UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MATHEUS MAIA ROQUE

INSERÇÃO DE LÓGICA DE PROGRAMAÇÃO NO ENSINO BÁSICO USANDO LINGUAGEM PYTHON E BIBLIOTECA PYGAME

FORTALEZA

2017

MATHEUS MAIA ROQUE

INSERÇÃO DE LÓGICA DE PROGRAMAÇÃO NO ENSINO BÁSICO USANDO LINGUAGEM PYTHON E BIBLIOTECA PYGAME

Monografia submetida ao Curso de Graduação em Engenharia Elétrica da Universidade Federal do Ceará como parte dos requisitos para a obtenção do título de Bacharel em Engenharia Elétrica. Área de concentração: Ciência, Tecnologia e Educação.

Orientador: Prof. Dr. Henrique Antunes Cunha Júnior

Coorientador: Dr. Cesar Rodrigues Fernandes

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R69i Roque, Matheus Maia.
Inserção de Lógica de Programação no Ensino Básico usando Linguagem Python e Biblioteca Pygame / Matheus Maia Roque. – 2017.
122 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2017.
Orientação: Prof. Dr. Henrique Antunes Cunha Júnior. Coorientação:
Prof. Dr. César Rodrigues Fernandes.

1. Educação. 2. Programação. 3. Python. 4. Pygame. I. Título.

CDD 621.3

MATHEUS MAIA ROQUE

INSERÇÃO DE LÓGICA DE PROGRAMAÇÃO NO ENSINO BÁSICO USANDO LINGUAGEM PYTHON E BIBLIOTECA PYGAME

Monografia submetida ao Curso de Graduação em Engenharia Elétrica da Universidade Federal do Ceará como parte dos requisitos para a obtenção do título de Bacharel em Engenharia Elétrica. Área de concentração: Ciência, Tecnologia e Educação.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____

Prof. Dr. Henrique Antunes Cunha Júnior (Orientador)

Universidade Federal do Ceará (UFC)

_____

Dr. Cesar Rodrigues Fernandes (Coorientador)
Universidade Federal do Ceará (UFC)

_____

Me. Liduina Lopes Alves
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Aos meus pais Lúcia Maia Roque e Antônio Roque do Carmo, que tanto lutaram e permaneceram de cabeça erguida com foco em dar a melhor educação e caráter aos seus dois filhos.

Ao meu irmão e grande amigo André Luiz Maia Roque que me mostrou da forma mais difícil o verdadeiro significado da vida e que desistir não é uma opção.

A meu orientador Dr. Henrique Antunes Cunha Júnior que me aceitou de braços abertos nesse projeto e me mostrou outra face da engenharia.

Aos meus amigos de faculdade José Roberto e Jeymyson Alves por dividirem comigo muito mais que o saber acadêmico, mas o crescimento pessoal e emocional que adquirimos durante essa grande jornada.

À Pró-Reitoria de Assuntos Estudantis (PRAE) por me oferecer condições de permanência na universidade.

À Seara da Ciência, em especial ao Prof. Dr. César Rodrigues Fernandes, pelo apoio no desenvolvimento desse projeto.

Ao meu namorado Jefferson Cardoso e meu grande amigo Marcos Carneiro por permanecerem presentes nesse ano difícil, dando-me direcionamentos e motivação para completar esse projeto.

A todos os amigos de Limoeiro, do intercâmbio, da residência universitária e da universidade que de alguma forma me deram suporte e dividiram momentos de tristeza e alegria, em especial Washington, Riccieli, Brenda, Abmael, Luana, Marcelo, Raviel, Minerva, Malena, Gabriela e Pedro.

“Happiness can be found, even in the darkest of times, if one only remembers to turn on the light.”

Alvo Percival

RESUMO

O objetivo é levantar questões a respeito de metodologias utilizadas para o ensino de linguagem e lógica de programação durante a educação básica bem como da insuficiência/inexistência dessa prática, resultando em futuros profissionais e estudantes de graduação com deficiência no desenvolvimento dessa habilidade. Para efetivar a justificativa, levantou-se um estudo bibliográfico através de artigos que tratam do desenvolvimento de projetos que visam à inserção de programação durante a educação básica, bem como a realização de uma pesquisa com alunos de diferentes universidades a respeito das dificuldades em desenvolver a habilidade de lógica de programação. O projeto traz ainda um estudo de caso no qual se utilizou a linguagem de programação *Python* e a temática *Arcade Games*, sendo este aplicado no espaço Seara da Ciência da Universidade Federal do Ceará com alunos pertencentes ao ensino médio de escolas públicas da cidade de Fortaleza/CE. O estudo de caso objetiva o desenvolvimento de uma metodologia de ensino de lógica de programação e a avaliação da efetividade do projeto no aprendizado dos estudantes participantes. Comprovou-se através do trabalho que a metodologia desenvolvida teve um grau de satisfação elevada entre os alunos participantes, e que a linguagem *Python* juntamente com a biblioteca *Pygame* conseguiram despertar no aluno o interesse por programação e estimular a criatividade dos mesmos para resolver problemas através da lógica computacional. O trabalho resultou ainda na criação e desenvolvimento de uma apostila didática que pode ser utilizada na aplicação de trabalhos futuros.

Palavras-chave: Educação, Programação, *Python*, *Pygame*.

ABSTRACT

This project aims to open questions about the methodologies used to teach programming language and logic for basic education students and the failure/lack of this practice, resulting in professionals and graduated students with problems to develop this ability. The justification can be effected by the bibliographic study done by articles which talk about development of projects aimed the insertion of programming in basic education, as well as the realization of a search with students from different universities. This search tries to show the difficult to develop programming and logic ability in graduating students. This project brings a case study using programming language Python and the theme Arcade Games, it was applied in the Seara da Ciência of the Federal University of Ceará (UFC). It was chosen high school students from Fortaleza/CE - Brazil to participate of this project. The case study goals are the development of a methodology teaching in programming logic and the evaluation of the effectiveness in the learning process in the students. It was proven that this project had a high satisfactory among the participating students, and the Python language together with the Pygame Library could bring the interest of the student in learn programming logic and awaking the creativity to solve different problems. This project had result in a elaboration of a didactic handout that can be used in future projects.

Keywords: Education. Programming. Python, Pygame.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CCB	Code Club Brasil
IDLE	Integrated Development and Learning Environment
OBI	Olimpíada Brasileira de Informática
ONG	Organização não Governamental
PSA	Python Software Activity
PSF	Python Software Foundation
RGB	Red, Green, Blue
SBC	Sociedade Brasileira de Computação
UFC	Universidade Federal do Ceará

LISTA DE SÍMBOLOS

%	Porcentagem
∑	Somatória

SUMÁRIO

1 INTRODUÇÃO

A educação básica sempre foi alvo de pesquisas e projetos, alguns destes com o intuito de elaborar uma ementa de disciplinas que viessem a fornecer os conhecimentos mínimos necessários a crianças e adolescentes para que consigam vivenciar a sociedade e o mercado de trabalho nos quais estão inseridos.

No entanto, pesquisadores como Schäfer et al.(2011), Rebouças et al.(2010) e Garcia et al.(2008) defendem que o ensino das disciplinas ditas como básicas (matemática, português, história, física, etc.) não são suficientes para que alunos consigam se adequar ao cenário do mercado atual. O ensino de linguagem e lógica de programação deve ser introduzido no meio educacional de jovens e crianças ainda no período do ensino fundamental ou médio, não privando esse conhecimento para alunos oriundos de graduações ou técnicos.

De acordo com Oliveira et al. (2014) que elaborou um projeto para ensinar linguagem de programação *Scratch* para alunos de ensino fundamental " [...] a computação é sim importante no ensino fundamental e sempre que possível deve ser inserido no currículo das escolas", o autor enfatiza essa importância através do acompanhamento que realizou no progresso educacional das crianças participantes do projeto.

Propomos através desse trabalho, uma discussão a respeito do ensino de programação para alunos da rede básica de educação, listando visões de pesquisadores a respeito do tema. Um paralelismo entre essa disciplina e a importância da mesma nos cursos de graduação (em especial engenharia elétrica e computação) será realizado, além do estudo de caso aplicado na *Seara da Ciência* utilizando linguagem *Python* para alunos do ensino médio.

1.1 Justificativa

O ensino de programação vem por muito tempo sendo negligenciado durante o período escolar e permanecendo exclusivo para graduações e técnicos de áreas voltadas para engenharia e tecnologia. No entanto, o crescimento tecnológico do Brasil e do mundo vem exigindo de seus profissionais, cada vez mais, um conhecimento na área de desenvolvimento de jogos, aplicativos *móbile*, sistemas, aplicações *web*, dentre outros; estes conhecimentos são possíveis através do estudo e aplicação de linguagem e lógica de programação. De acordo com o site de empregos *Cothe*, em entrevista à BBC Brasil, em 2015, o ramo de tecnologia é hoje, no Brasil o setor que oferece maiores oportunidades de empregos (COSTAS, 2015).

A grande demanda por profissionais com habilidades de programação e desenvolvimento nada mais é que um reflexo do *déficit* de profissionais nessa área. Países como Estados Unidos e Austrália também sofrem com esse problema. De acordo com Burdon (2015) o governo australiano, visando o aumento do número de profissionais, estipulou o ensino obrigatório de programação para alunos a partir dos 10 anos de idade. Já em Nova York, uma nova lei deve, até 2018, tornar obrigatório o ensino dessa disciplina na rede básica da cidade (FLORENZANO, 2015).

No Brasil o ensino de programação ainda não é obrigatório. A Sociedade Brasileira de Computação (SBC) defende o ensino da mesma no período da educação básica, no entanto essa prática fica a mercê de projetos de extensão oferecidos por universidade ou através de Organizações não Governamentais (ONGs) (OLIVEIRA et al. 2014).

> Hoje em dia conhecer tecnologia não é o bastante, o grande diferencial é saber a linguagem de programação. Existe um movimento mundial a favor do ensino de programação para crianças e adolescentes em fase escolar, por conta dos benefícios que traz a vida acadêmica do aluno, como também, no desenvolvimento pessoal e ainda pensando no futuro, para atender a demanda de profissionais qualificados nessa área. (HAPPY CODE, 2016).

A Happy Code, grupo de suporte às escolas que visam o ensino de programação e robótica, defende alguns benefícios que a prática de usar essa ciência no ensino básico pode oferecer:

- Aproximação de escolas tradicionais com a tecnologia;
- Ajuda o aluno a desenvolver novas habilidades e competências, como o pensamento lógico e solução de problemas;
- Democratização do ensino de programação no Brasil.

Pode-se ainda afirmar, baseado na vivência pessoal do autor, que muitos alunos oriundos de escolas públicas não possuem acesso a esse tipo de conhecimento. O uso de computadores nas escolas é casual e utilizado somente para introduzir os alunos a essa máquina: acesso às páginas de internet, fundamentos de hardware e criações de perfis em redes sociais. Estudantes da rede pública de ensino básico obtém o conhecimento de lógica de programação somente ao entrarem em cursos de nível técnico ou superior; assim, ao se depararem com essa ciência em um nível avançado, acabam por não usufruírem efetivamente do poder que a programação poderia ofertar.

O conhecimento da possível importância do ensino de programação para alunos do ensino fundamental e médio, as estatísticas de déficit de profissionais na área de tecnologia

da informação e as vivências pessoais do autor motivaram a realização desse trabalho de pesquisa, o qual visa realizar um estudo a respeito da importância dessa prática para alunos da rede básica.

1.2 Objetivos do trabalho

O objetivo principal deste trabalho é a proposição de um debate a respeito da importância de se lecionar e aprender linguagem e lógica de programação durante o ensino básico. Atrelado a esse assunto, discutir-se-á a insuficiência (ou a inexistência) do ensino dessa disciplina nas escolas de ensino fundamental e médio do Brasil.

Ainda será realizado um estudo sobre as principais linguagens de programação lecionadas às crianças e jovens a fim de se desenvolver uma metodologia de ensino a ser aplicada a alunos oriundos da rede pública.

Através do estudo de caso, objetivos secundários foram gerados: verificar, através da prática, se alunos do ensino básico possuem capacidades para apreender e realizar atividades que envolvam o uso de programação, e discorrer da metodologia utilizada para esse evento, verificando através de *feedback* dos alunos a efetividade do projeto.

Para enfatizar as proposições, o trabalho objetiva ainda a realização de uma pesquisa entre alunos universitários para se apurar o quanto a programação está presente em seus cursos de nível superior e quantos deles tiveram um contato prévio, ao ingresso na universidade, com essa ciência.

1.3 Estrutura do trabalho

O capítulo inicial desse trabalho faz um pequeno relato da problemática estudada nas páginas seguintes, justificando a escolha do tema bem como os objetivos a serem alcançados.

O capítulo 2 exemplificará os conceitos usados para designar linguagem de programação, suas diversidades e aprofundará nas características da linguagem escolhida para elaboração do estudo de caso, a linguagem *Python*.

Após a conceituação de linguagem de programação, o Capítulo 3 trará um estudo bibliográfico sobre metodologias aplicadas para o ensino de programação no ensino básico e opiniões de autores favoráveis e contrários ao ensino dessa disciplina no ensino fundamental e médio.

No capítulo 4 foi realizada uma averiguação a respeito da importância da programação no nível superior e as dificuldades que os alunos possuem nessa ciência. Realizou-se ainda uma pesquisa com alunos de diferentes universidades sobre o contato deles com programação durante a educação básica e a coleta de opinião sobre a temática proposta nesse projeto.

O capítulo 5 é dedicado ao estudo de caso. Contém detalhes a respeito da elaboração e execução do projeto intitulado *Curso Introdutório de Programação para Alunos do Ensino Médio*, mostrando ainda os resultados obtidos com o projeto, o perfil socioeconômico dos alunos participantes e o *feedback* oferecido por eles a respeito do período em que participaram da ação.

No final, são realizadas as conclusões do trabalho, propostas de trabalhos futuros, referências teóricas e apêndices que trazem mais detalhes a respeito do projeto aplicado como estudo de caso.

2 LINGUAGEM DE PROGRAMAÇÃO: UMA ÊNFASE EM LINGUAGEM *PYTHON*

Para a realização do estudo de caso proposto nesse trabalho, fez-se necessária a escolha de uma linguagem de programação e um tema ao qual o projeto seria direcionado. Para isso, foi feito um estudo a respeito das principais linguagens e as mais usadas no âmbito educacional, bem como as áreas para o qual cada uma delas tem maior credibilidade, foi feita. Após esse estudo optou-se pela escolha da linguagem *Python* e o tema *Arcade Games*.

Essa seção vem então com o objetivo de definir e exemplificar programação além de adentrar um pouco nos conceitos e estrutura da linguagem escolhida para este projeto.

2.1 Algoritmos

Computadores são por muitas vezes consideradas máquinas inteligentes, com capacidades de executar inúmeras tarefas e automatizar atividades para o ser humano. No entanto Kochan (2005, p.5) afirma que o computador é na verdade uma máquina “burra”, visto que este só executa funções que foram previamente ditas de uma forma bastante exemplificada para ser executada. O que na verdade essa máquina consegue fazer, por capacidade própria, é somar dois números ou testar quando um número é ou não igual ao zero.

Para resolver determinado problema, com a ajuda de um computador, é necessário escrever um conjunto de instruções de forma simples e estruturada. Um programa de computador é então uma coleção de instruções que determinará as etapas necessárias a serem aplicadas para obter um resultado esperado. Esse conjunto de instruções é o que conhecemos por Algoritmo que pode ser definido como “[...] um texto contendo comandos (instruções) que devem ser executados numa determinada ordem. Esse texto em si não nos interessa, mas, sim, seu significado, ou seja, aquilo que ele representa” (GUIMARÃES; LAGES, 1994).

Com o algoritmo em mãos é então possível transcrevê-lo para o computador numa linguagem que possa ser por ele entendida. Algumas linguagens de programação foram criadas para fazer essa tradução tais como *Visual Basic*, *Java*, *C*, *C++* ou mesmo a *Python.*

2.2 História das linguagens de programação

Quando computadores foram inicialmente desenvolvidos, a única maneira de comunicação entre o programador e a máquina era em forma de linguagem binária, sistema representado por 0's e 1's. Esta linguagem é a única entendida pela máquina, tendo assim o programador que se adequar para "falar" com o computador, transformando seu algoritmo em conjunto de dados formado por sequências binárias.

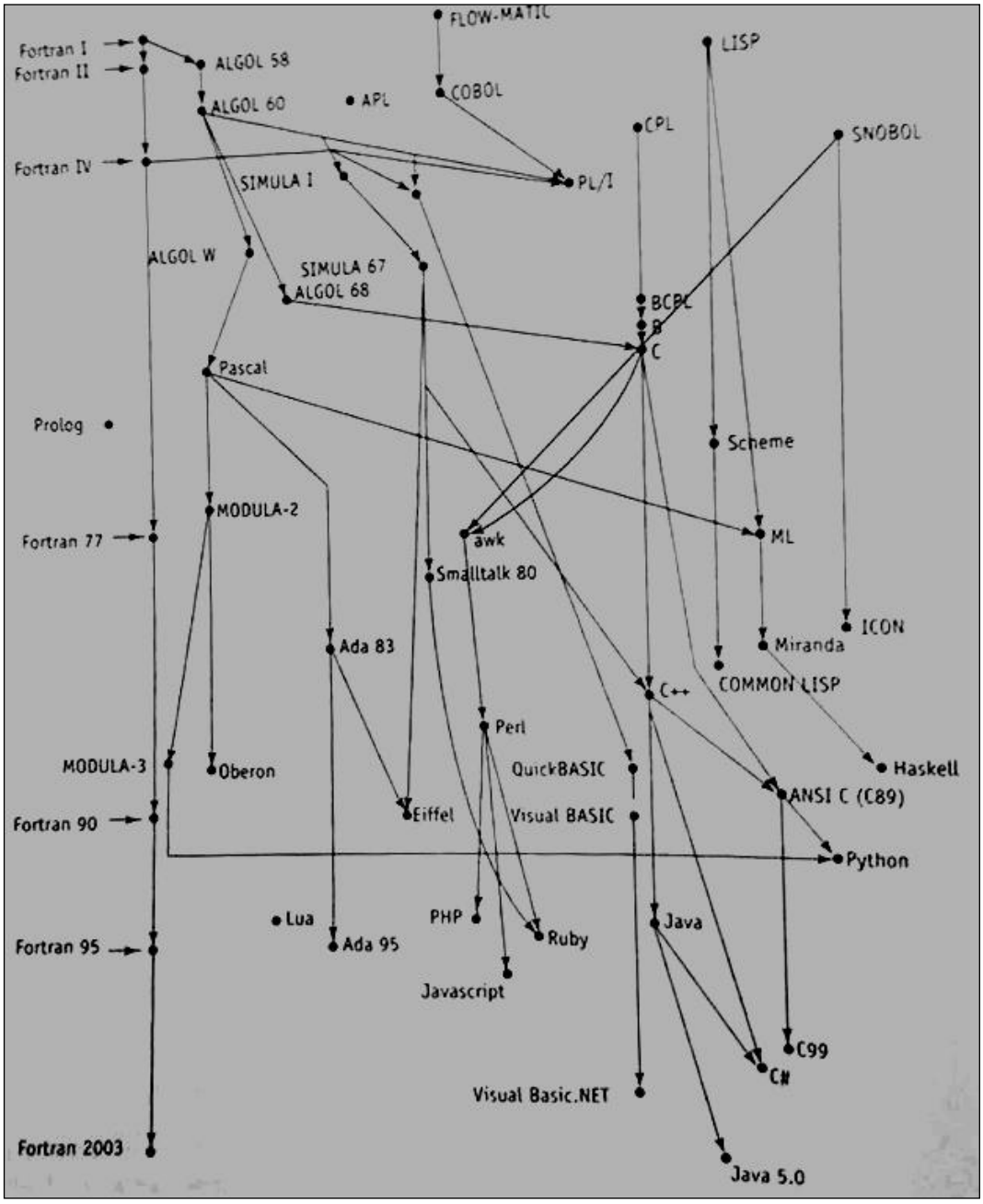
Logo mais, com o intuito de reduzir as grandes e árduas sequências de comando no sistema de dois algarismos, foi criada a linguagem *assembly*. Com ela o programador poderia carregar suas informações através de nomes simbólicos que representavam comandos ou mesmo endereços de memórias (KOCHAN, 2005, p.6). O programa *assembler* ficou então encarregado de transcrever a linguagem *assembly* para a linguagem binária da máquina.

Devido à *assembly* ser uma linguagem próxima da linguagem de máquina, trabalhando com *bits*, *bytes* e conteúdo de memória, ela é considerada uma linguagem de baixo nível. Willrich (2000) afirma que as linguagens de alto nível vieram apenas em meados das décadas de 50 e 60 sendo FORTRAN dita como a pioneira. Sua principal característica era a possibilidade de um programa escrito em seus termos, ser lido por qualquer máquina que suportasse a linguagem, não necessitando reformular o algoritmo cada vez que fosse utilizado por computadores diferentes (SEBESTA, 2011).

"Em comparação com a linguagem *Assembly*, a passagem de um programa escrito em linguagem de alto nível para o programa em linguagem de máquina é bem mais complexa. Para esta passagem são utilizados compiladores e *linkadores."* (WILLRICH, 2000). Os compiladores são então programas que transcrevem a linguagem de alto nível para linguagem própria de máquina.

Com o objetivo de facilitar projetos em suas respectivas áreas, novas linguagens de programação foram sendo desenvolvidas com o decorrer do tempo para suprir os desejos dos programadores. A Figura 1 a seguir mostra a genealogia das principais linguagens de programação de alto nível. Através da imagem é possível perceber que muitas das linguagens mais utilizadas hoje são derivadas da pioneira FORTRAN. A linguagem *Python,* por exemplo, possui características provenientes da MODULA e do C++.

Figura 1 - Genealogia das principais linguagens de programação de alto nível.



Fonte: Sebesta (2011, p.58)

2.3 Linguagem *Python*

Um dos dilemas enfrentados no ensino de programação para crianças e adolescentes é a respeito da capacidade destes de assimilar o conteúdo, visto que programação demanda conhecimentos de matemática, lógica e, em alguns casos, até mesmo da língua inglesa. Para demonstrar a eficiência do ensino dessa disciplina durante a educação básica, foi

desenvolvido um estudo de caso[1] utilizando uma das linguagens mehor conceituadas atualmente: a linguagem *Python.*

As subseções a seguir visam mostrar a história dessa linguagem, as vantagens de utilizá-la em relação às demais e algumas de suas características. Essa pesquisa teve sua importância para a escolha de uma linguagem que fosse acessível para os alunos e para o ministério das aulas.

2.3.1 História

A linguagem *Python* foi inicialmente pensada no início dos anos 80 através do holandês Guido Van Rossum quando este trabalhava na CWI (*Centrum Wiskunde & Informatica* (Centro de Matemática e Informática) em Amsterdã/Holanda. Guido participava do projeto AMOEBA, um sistema operacional *Microkernel*[2], e percebeu que alguns dos recursos que dispunha para a realização do projeto não eram de total eficiência além de levar horas de trabalho para a realização das atividades. Seus recursos estavam limitados entre as linguagens *C* e o *Shell Bourne* e ele precisava de algo intermediário. Foi então, em 1989, que o projeto *Python* teve início (MIND BENDING, 2014). De acordo com Oliveira (2010) o nome *Python* se deu pela afinidade que o criador da linguagem possuía com o grupo de comédia britânico *Monty Python* (também conhecidos como *The Pythons*).

Após ser transferido do grupo AMOEBA para o grupo Multimídia em 1991, Guido lançou a primeira versão do *Python* (20 de Fevereiro) denominada v0.9.0. Nessa primeira versão a linguagem já contava com classes, herança, tratamento de exceções, funções, sistemas de módulos e os tipos de dados nativos *list, dict, str,* e etc. (MIND BENDING, 2014).

Em Abril de 1995, Guido foi trabalhar na CNRI (*Corporation for National Research*) onde liderou um time do desenvolvimento de um sistema escrito puramente em *Python*. Esse mesmo time criou a primeira organização sobre *Python* a PSA (*Python Software Activity*) que se tornou um marco importante para o crescimento e difusão da linguagem através do sitio *python.org.* Neste período novas versões da linguagem foram lançadas conforme a

[1] O estudo de caso aqui tratado será mostradodo no capítulo 4 desse trabalho.
[2] Arquitetura de núcleo de um sistema operacional cujas funcionalidades são quase todas executadas fora do núcleo. Os processos se comunicam com um núcleo mínimo. (StackOverflow, 2016)

Tabela *1* abaixo.

Tabela 1 - Versões da Linguagem Python durante período CNRI.

Mês / Ano	*Versão*
Outubro de 1995	*1.3*
Outubro de 1996	*1.4*
Janeiro de 1998	*1.5*
Outubro de 1998	*1.5.1*
Abril de 1999	*1.5.2*
Setembro de 2000	*1.6*

Fonte: Mind Bending (2014)

Em 2001, já deixado a CNRI, os associados da licença *Python* fundaram a *Python Software Foundation* (PSF), cuja missão principal é "[...] promover, proteger e desenvolver a linguagem de programação *Python*, bem como dar suporte e facilitar o crescimento da diversidade e internacionalização da comunidade de programadores *Python."* (PYTHON SOFTWARE FOUNDATION, 2017).

Após a criação da fundação, todas as atualizações desde a versão 2.1 são feitas utilizando a PSF *License Agreement* (Licença de concordância PSF). Atualmente, o idealizador Guido trabalha para a empresa *Dropbox*, já tendo passado pela *Google* (2005).

2.3.2 Características

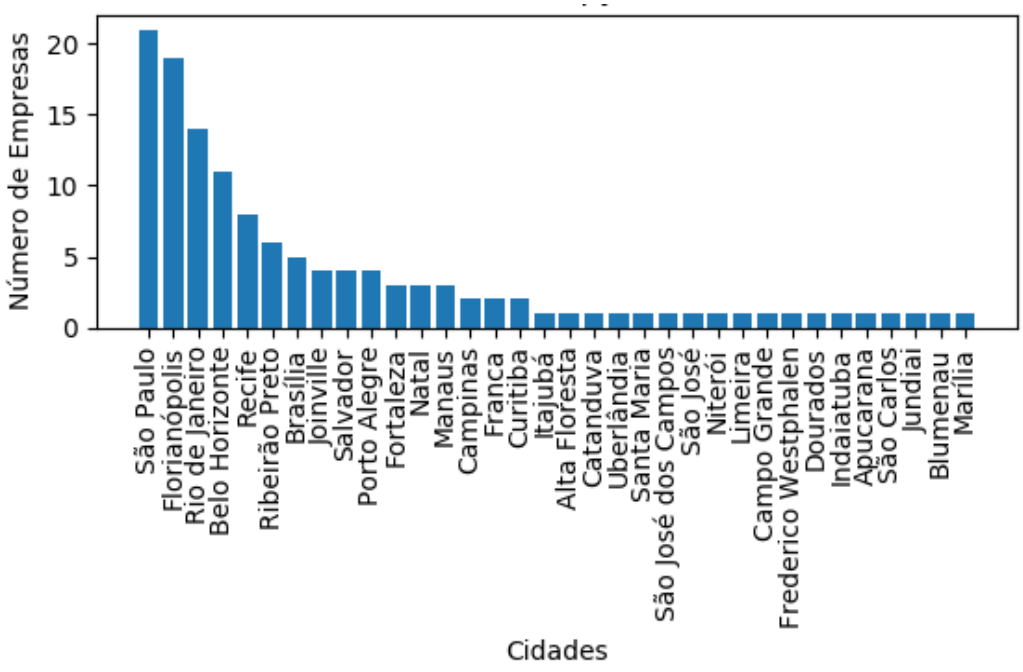
A escolha de uma linguagem de programação deve ser feita de forma criteriosa, analisando os objetivos do projeto bem como o público alvo que a utilizará. Mesmo que a linguagem de programação não seja o foco do projeto, como no presente caso, mas sim a aprendizagem de lógica usada na programação, a escolha é de total importância podendo facilitar ou prejudicar o entendimento da disciplina por parte dos usuários.

No estudo de caso proposto para esse trabalho, o público-alvo são alunos da rede básica, pertencentes ao ensino médio. Para isso a escolha da linguagem *Python* se deu através do estudo do trabalho descrito por Ramalho (1999). Em suas pesquisas este tipo de linguagem é uma das melhores escolhas para quem quer aprender a programar e ganhar a percepção de lógica exigida nessa área.

A principal característica analisada por Ramalho (1999) é o fato de a linguagem *Python* ser do tipo *scripting*, o que a torna mais simples em relação a linguagens copiladas como a *C++* ou *Java*. Outras linguagens do tipo *scripting* foram vistas como competidoras para a *Python*, por exemplo, *JavaScript* e *Pearl*. No entanto *JavaScript* tem suas aplicações voltadas quase que totalmente para a área de *web-sites* dinâmicos enquanto que *Pearl*, segundo o próprio criador, foi criada para agradar *hackers;* ambas características não são de interesse para o projeto que se pretende atender.

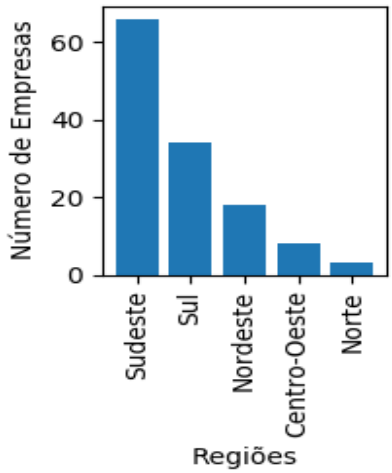
A linguagem *Python* é também uma linguagem recente com uma grande aderência no mercado de trabalho. Algumas empresas do Brasil já se enquadram nas chamadas empresas "pythônicas", ou seja, empresas que utilizam da linguagem *Python* para programar seus sistemas e projetos de desenvolvimento. A Figura 2 a seguir mostra o número de empresas "pythônicas" nas principais cidades do país, enquanto que a Figura 3 faz essa comparação com as regiões do Brasil.

Figura 2- Cidades com maior número de empresas "pythônicas".



Fonte: Python Brasil (2017)

Figura 3 - Regiões mais "pythônicas" no Brasil.



Fonte: Python Brasil (2017)

No estado do Ceará, estado onde o projeto será aplicado, existem três grandes empresas que trabalham com esse tipo de linguagem:

DEWAY: Empresa especialista na criação de programas para iPhone, iPad e Android.. A empresa foi fundada em março de 2009 por Edgar Matos e com o apoio e colaboração de Leonardo Jalk. Em setembro de 2011, Adriano Sousa, foi integrado ao quadro de sócios fortalecendo o crescimento e estrutura. (DEWAY, 2017)

FUNCEME: Fundação Cearense de Meteorologia e Recursos Hídricos que tem como finalidade difundir o conhecimento da geografia física do estado do Ceará. (FUNCEME, 2009)

MORPHUS: Especialistas em Segurança da Informação, uma empresa com reconhecimento nacional, desde 2002. Seus serviços estão voltados para a área de segurança ofensiva, segurança defensiva, riscos e educação. (MORPHUS, 2017)

Ainda que *Python* seja uma linguagem considerada nova no país, as expectativas para seu crescimento são altas devido às suas características e facilidade em trabalhar em diferentes sistemas operacionais (diferente de *C* e *Java*).

2.3.3 Escolha do ambiente de desenvolvimento: Wing IDLE

Na seção 2.3.2 foi constatado que a linguagem *Python* é do tipo *scripting* e não compilada, ou seja, não é necessário um programa “compilador” para transcrever a linguagem de alto-nível para a linguagem de máquina. No entanto, para realizar essa tradução, faz-se

necessário o uso de um interpretador que realizará a tradução linha a linha; assim o programa é utilizado à medida que vai sendo traduzido. (MULLER, 2009)

Os interpretadores na linguagem *Python* são chamados IDLE (*Integrated Development and Learning Environment*, Ambiente Integrado de Desenvolvimento e Aprendizado) que, além de serem responsáveis pela tradução da linguagem de alto-nível, também possuem a intenção de ajudar no aprendizado da linguagem em questão. Um IDLE possui funções como correção de sintaxe, sugestão de comandos e de possíveis erros nas linhas programadas, auxiliando dessa forma o usuário que está aprendendo e se integrando às regras "gramaticais" da linguagem.

Tão importante como a escolha da linguagem de programação, a escolha do ambiente de desenvolvimento é fundamental no auxílio de aprendizado do programador. Amaral (2012) comenta sobre os principais e mais usuais IDLE para quem esta iniciando a programação em *Python* e afirma que a escolha de um IDLE ideal é impossível, visto que a alternativa depende de inúmeros fatores tais como o nível do programador, o sistema operacional usado, o tipo de aplicação, etc. No entanto, entre os mais usados foram citados: *PyCharm, WingIDE, Komodo IDE, Eclipse + PyDev* e *Geany.*

Para o projeto em questão, optou-se pelo uso do *WingIDE*, devido a sua simplicidade e facilidade na obtenção para qualquer sistema operacional.

2.3.4 Biblioteca Pygame

As seções acima mostraram as principais características da linguagem *Python*, suas qualidades em relação às demais linguagens (acessíveis para o ensino de programação a iniciantes) e as vantagens de se usar um IDLE como o Wing IDLE.

No entanto, apenas a escolha de uma linguagem adequada e um ambiente de desenvolvimento que facilite o aprendizado não é suficiente. É necessário ainda trazer até o jovem uma temática de projeto que os motivem a usarem a criatividade e a buscarem conhecimentos no aprendizado dessa nova (para eles) ciência.

Segundo Dinello (2004), por meio das atividades lúdicas:

> "As crianças manifestam, com evidência, uma aprendizagem de habilidades, transformam sua agressividade em outras relações criativas, crescem em imaginação e se socializam, melhorando o vocabulário e se tornando independentes".

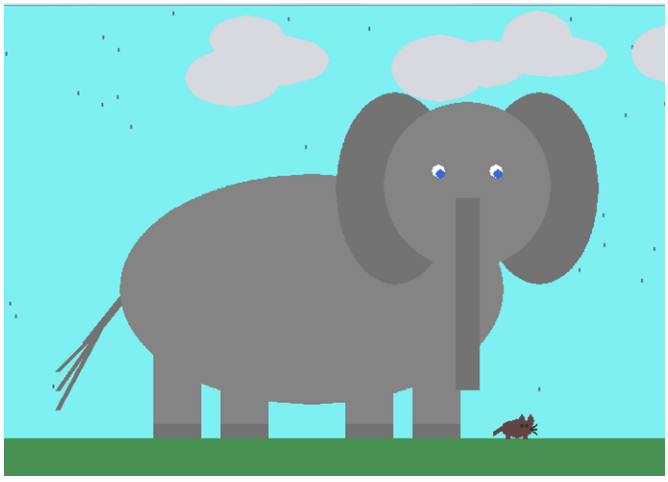
Dentre as principais atividades lúdicas defendidas por Neto (2001), os jogos digitais são, sem dúvidas, de vital importância para que crianças e jovens tornem-se seres independentes, capazes de se expressarem, realizando experiências e descobertas. Outros pesquisadores, como Silveira (1998), ainda defendem que os jogos educacionais computadorizados são elaborados para divertir os alunos e prender sua atenção, auxiliando no aprendizado de conceitos.

Reconhecendo então que jogos com teor educacional podem auxiliar alunos no aprendizado e ainda prender a atenção deles, optou-se por utilizar a biblioteca *Pygame*, disponibilizada gratuitamente, como meio de se introduzir lógica de programação para os alunos do ensino médio, alunos estes sujeitos do estudo de caso aqui proposto.

A biblioteca *Pygame* é um módulo *Python* que oferece funcionalidades para a programação gráfica, em especial jogos. Ela provê facilidades tais como *Sprites* (movimentação gráfica), *Render Groups*, deteção de colisão básica e muito mais, sendo ainda multi plataforma, funcionando em *Linux*, *Windows*, *MacOS*, e outros.

A Figura *4* abaixo demonstra um exemplo de projeto que pode ser realizado através dessa biblioteca juntamente com a linguagem *Python*.

Figura 4 - Exemplo do uso da biblioteca *Pygame*



Fonte: Próprio autor (2016).

Ainda é possível a elaboração de imagens em movimento, bem como o controle de objetos usando o *mouse* e o teclado do computador. Dessa forma, através da junção dessas características, o aluno iniciante na programação poderá usar da sua criatividade e do seu poder de raciocínio lógico para desenvolver seus próprios jogos ou animações.

3 O ENSINO DE PROGRAMAÇÃO NA REDE BÁSICA: METODOLOGIAS E DISCURSÕES

Diferente de países como Inglaterra, Estados Unidos e Austrália, o Brasil ainda não possui uma lei ou projeto de lei que torne o ensino de programação nas escolas da rede básica como obrigatório (ARAÚJO ET AL., 2015). No entanto, já percebendo a importância dessa disciplina na formação da criança e adolescente, muitas universidades e ONGs estão desenvolvendo e aplicando projetos para levar esse conhecimento a jovens usuários.

Nessa seção serão apontados alguns projetos já desenvolvidos e os resultados obtidos, focando na metodologia apresentada e na capacidade desses alunos de absorverem o conteúdo de lógica de programação. Esse estudo se torna importante para verificar possíveis erros de forma prévia e evitar cometê-los de igual maneira no estudo de caso feito aqui. Além desse embasamento teórico, serão analisadas opiniões de especialistas do ramo da educação que condenam essa aproximação da criança com o mundo da informática a fim de argumentar as condições defendidas.

3.1 Metodologias aplicadas para o ensino de programação na rede básica

Ensinar programação para alunos da rede básica não é uma tarefa fácil; é algo novo e que muitas vezes assusta o aluno em seu primeiro contato. Dessa forma, diversas metodologias são desenvolvidas com o objetivo de melhor levar esse conteúdo adiante. A busca pela linguagem de programação ideal, aulas presenciais ou virtuais, com computador ou *unplugged* são as principais questões levantadas.

Oliveira et. al (2013) desenvolveram um trabalho através de um projeto de extensão na Universidade Federal da Paraíba (UFPB) sobre o ensino de programação para alunos da rede estadual com o objetivo de incentivar essas escolas e alunos a participarem da OBI (Olimpíada Brasileira de Informática). Usou-se nesse projeto a metodologia EAD (Educação à Distância) através da plataforma *Moodle* (*Modular Object-Oriented Dynamic Learning*, Módulo de Aprendizado Dinâmico Orientado a Objeto) onde o material de ensino é disposto *online* e os alunos são autodidatas. A linguagem escolhida para a realização desse projeto foi a linguagem *Python* focando em conteúdos teóricos e menos práticos, visto que o projeto tende a se adaptar ao conteúdo das provas da OBI. O ensino através de EAD apresentou-se como algo desafiador, visto que o aluno é responsável pelo seu próprio

desenvolvimento, mas como vantagem possui a inclusão e consegue abranger um maior público.

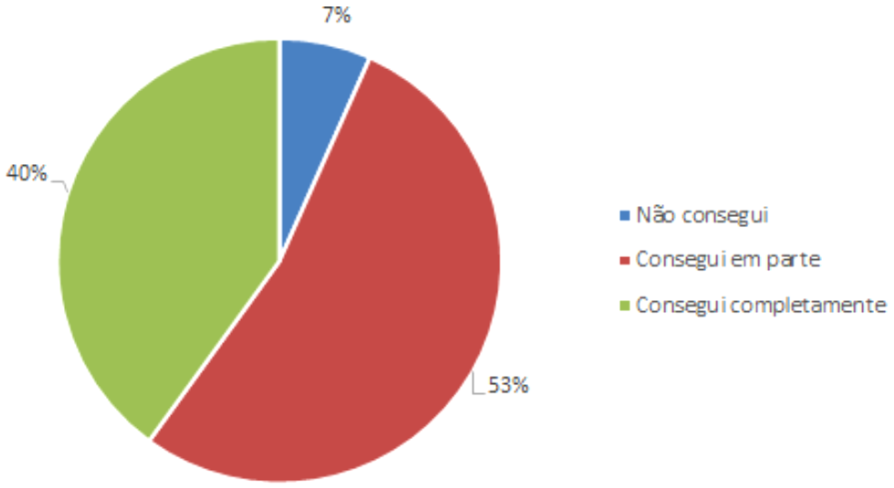
Existem no Brasil alguns projetos de maior abrangência e que já atuam em maior período de tempo. É o caso, por exemplo, do *Code Club Brasil* (CCB) que teve início na Inglaterra em 2012 e chegou ao Brasil em meados de 2013. O projeto é um intermediador entre voluntários e grupos de alunos que querem aprender programação com foco em criação de jogos, animações e *websites*. Além de fazer a ligação entre monitores voluntários que auxiliarão as crianças no projeto, o CCB ainda fornece material didático de qualidade e canais *online* de suporte (CODE CLUB BRASIL, 2017). O projeto torna-se então de grande utilidade para pesquisadores que querem desenvolver projetos de inclusão de programação e que não podem, por algum motivo, desenvolver o próprio curso e metodologia.

Muitos pesquisadores que desenvolvem projetos nessa área preferem a inserção através de aulas presenciais, onde existam monitores para facilitar a disseminação do conteúdo e auxiliar na tiragem de dúvidas. Esse modelo aplica-se ao estudo de caso realizado por Garlet et. al (2016) onde, baseado no plano de estudo realizado pela SBC, desenvolveu um conjunto de aulas para alunos do 7º ao 9º ano do ensino fundamental de duas escolas estaduais da região do Alto Médio Uruguai (RS). Nessa iniciativa os pesquisadores usaram como ferramenta a linguagem de programação *VisuAlg* por ela ter sido criada para fins educacionais e ser escrito na língua portuguesa. O projeto apresentou um alto nível de desistência justificado por ter sido escolhido o turno noturno para aplicação das aulas presenciais. No entanto, percebeu-se um alto nível de rendimento nos alunos que permaneceram até o final do projeto, mostrando, segundo os autores , a eficiência da linguagem escolhida e que alunos dessa faixa etária conseguiram absorver o conteúdo proposto.

Por outro lado, pesquisas realizadas por Leal (2014) mostram a dificuldade de alunos assimilarem o conteúdo de programação quando entram em contato direto com as linguagens de programação sem antes serem motivados e levados a conhecer a lógica dos algoritmos com situações físicas. Utilizando do espaço e da estrutura do Instituto Federal de Educação do Mato Grosso (IFMT) o pesquisador desenvolveu um projeto para alunos do ensino médio integrado desse educandário que visa integrar o aluno no processo de criação de lógica de programação. Usou-se a linguagem *C++* como apoio, e como principal atividade a criação de jogos envolvendo objetos físicos (como bola, cordas, garrafas, etc.) que possuíssem um conjunto de regras e sequência de passos, similar a um algoritmo de computador. Dessa forma foi possível fazer a concordância entre as etapas físicas e o digital quando se introduziu

a linguagem de programação. Como resultado, percebeu-se um aumento de 17,42% na média desses alunos na disciplina de introdução a programação, o que foi considerado pelo autor um fator que agrega valor ao método. A Figura 5 abaixo mostra o impacto que essa prática teve na visão de mundo de alguns alunos, visto que grande parte dos participantes conseguiu observar padrões de programação na elaboração dos jogos.

Figura 5 - Identificação dos padrões de programação na participação e criação dos jogos.



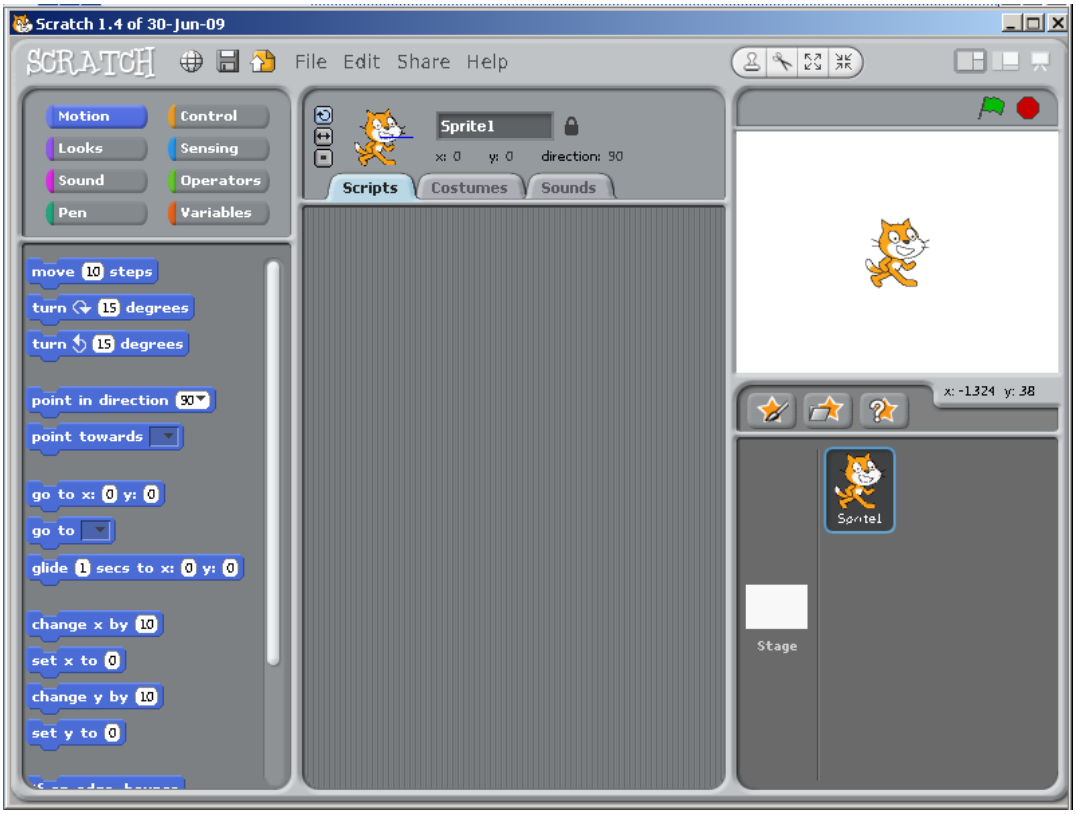
Fonte: Leal (2014)

Dentre as metodologias descritas em trabalhos e projetos analisados, o uso da linguagem *Scratch* é sem dúvida a mais utilizada por programadores como ferramenta de inserção de lógica de programação para crianças do ensino fundamental. A linguagem é entendida como:

> ”[...] Uma linguagem de programação e uma comunidade online onde crianças podem programar e compartilhar mídia interativa tal como estórias, jogos e animações com pessoas do mundo inteiro. As crianças podem criar com Scratch, podem aprender razões sistemáticas. a pensar de forma criativa e trabalhar em grupo.” (SCRATCH, 2017. Tradução nossa)

Características como o uso de comandos em forma de blocos, interface ilustrada e ser uma linguagem de alto-nível torna o *Scratch* uma ferramenta ideal para crianças que ainda não desenvolveram habilidades matemáticas suficientes para utilizar uma linguagem com uso de códigos escritos como é o caso do *Python*, *C++* ou *Java*. A Figura 6 a seguir, retirada do site oficial da linguagem, mostra a interface e o modelo de blocos.

Figura 6 - Interface da linguagem Scratch utilizando comando em blocos.



Fonte: Linux Journal (2010)

Nascimento (2015) desenvolve um projeto para alunos do 6º ano da rede pública da cidade de Rorainópolis - RR usando essa linguagem interativa de programação. Foi notória a efetividade dos alunos, mesmo nessa faixa etária, em aprender lógica de programação e o crescimento que os mesmos obtiveram em disciplinas paralelas como matemática e raciocínio lógico. Outros autores como Castro et. al (2017), Beer (2010) e Batista et. al (2015) também realizaram projetos e discutiram o uso dessa linguagem no ensino de lógica para estudantes do ensino fundamental, sempre ressaltando o entusiasmo e facilidade que os alunos obtém pelo *Scratch* devido a sua forma interativa e visual de ser programado.

Diferente de todas as metodologias já citadas, ensinar programação de forma *unplugged* (desconectada) é sem dúvida uma das mais criativas e inovadoras. O termo *unplugged* refere-se ao conceito de que não é necessário o uso de uma máquina para poder entender como ela funciona (BELL et. al, 1998). O livro *Computer Science Unplugged... off-line activities and games for all ages* (Ciência da Computação desconectada.. Jogos e atividades *off-line* para todas as idades) escrito por Bell et. al (1998) traz uma série de atividades voltadas para um público de ensino fundamental ou superior que remetem a ciência da computação mas sem usar o computador, apenas usando jogos lúdicos e atividades

manuais. Essa metodologia é também defendida no projeto “Tecnologias para a Aprendizagem Criativa” criado pelo Prof. Paulo Adriano Ferrari, que justifica através do aprendizado por trabalho em grupo e exploração:

> ”Tal abordagem favoreceu um aprender pela exploração, ao invés de limitar-se somente ao um aprender por instrução: ao enfrentar o desafio, aprende-se ao mesmo tempo em que se desbravam novos conhecimentos. Vai-se às vezes por saltos, às vezes por fases numa exploração despreocupada em busca da resolução do desafio e no desenvolvimento de competências e habilidades. Privilegia-se o trabalho em grupo, colaborativo. Várias cabeças, não só pensam mais que uma: criam, constroem e partilham conhecimentos, habilidades e aprendizagens multifacetadas.” (FERRARI, 2017)

Através desse estudo é possível perceber que a escolha da metodologia ideal a ser aplicada em um projeto, com fins de despertar em alunos da faixa infanto-juvenil a lógica de programação, deve ser realizada com base não apenas no grau de instrução dos alunos ou professor aplicador, mas também na estrutura material e no tempo disponível para o trabalho. Sem dúvidas o método *unplugged* contribui para uma inserção prévia e ajuda os novos usuários a encontrarem um significado real por trás das linhas de códigos, e a utilização de jogos e efeitos gráficos contribuem de forma significativa para despertar no aluno a curiosidade e o interesse por esta área.

3.2 Prejuízos da programação no desenvolvimento infantil

Na seção anterior foi possível se familiarizar com alguns projetos e metodologias que têm como objetivo a inserção de programação para jovens aprendizes. Os autores e trabalhos citados defendem veementemente que essa ciência deve ser inserida durante o ensino básico e que seu aprendizado traz não apenas conhecimento de computação, mas alavanca as percepções lógicas e as habilidades dos alunos nas demais disciplinas escolares.

Mesmo com estudos publicados que mostram essas vantagens, existem ainda grupos de pesquisadores e de profissionais da área de tecnologia que veem essa prática como desnecessária e muitas vezes prejudicial para o crescimento da criança e do adolescente.

Linus Torvalds, criador do sistema operacional *Linux* afirmou em uma entrevista que: “Na verdade eu não acredito que todos devam necessariamente tentar aprender a programar [...]. Não é como saber ler e escrever e fazer contas básicas.” (LOVE, 2014). Diz ainda que programação é uma ciência que requer alto nível de habilidades lógicas, o que para

muitos não é necessário; no entanto acredita-se que todos devam ter ao menos um contato inicial para tomarem ciência de que aquela área não lhe é cabível.

Quando levamos a pergunta para alguns profissionais da área de TI, por exemplo, a opinião é semelhante à de Linus. Segundo Geraldes (2014) muitos profissionais dessa área acreditam que lógica de programação é uma habilidade avançada e que requer raciocínio lógico apurado e capacidade de resolver problemas com alto grau de complexidade, características que, para eles, nem todo mundo possui.

Alguns educadores pedagogos também se dizem contrários a essa prática. Setzer (2012) diz que a introdução precoce nas vidas de crianças e jovens podem trazer danos mentais, prejudicando a infância dos mesmos. Diz ainda que essa ciência leva a desenvolver um pensamento que deveria ser adquirido apenas na fase adulta, o que acarreta numa perda de infantilidade que é de extrema necessidade para o desenvolvimento do ser humano. Alguns educadores acreditam na teoria de que categoricamente o ensino de programação prejudica a infância, por retirar dela a oportunidade de uma criança se socializar com outras crianças através de brincadeiras e jogos próprios dessa fase.

3.3 Comparação crítica

Os principais argumentos utilizados por educadores e profissionais da área de tecnologia para justificar os prejuízos de se ensinar programação para alunos da rede básica estão sustentados pela complexidade da disciplina e da perda de infantilidade e convívio social que o computador causa.

No entanto, os trabalhos mostrados na seção 3.1 mostram diferentes maneiras de facilitar a inserção dessa ciência na vida educacional de crianças e jovens. Linguagens como a *Scratch* e *VisuAlg* são exemplos de linguagens com interfaces fáceis e interativas que dão ao aluno condições de aprender a lógica de programação sem necessidade de possuir grandes conhecimentos matemáticos. A seção ainda traz exemplos de trabalhos *unplugged* que criam no aluno o interesse pelo aprendizado e que ajudam a ver o significado real por traz de uma sequência de comandos (algoritmos).

Geraldes (2014) rebate a problemática da perca da infantilidade com a aproximação da criança com o computador. Afirma que atualmente estamos cercados pelo mundo digital, já nascemos inseridos nesse mundo e que o não conhecimento de programação nos torna meros espectadores. Para que nos tornemos mais ativos e cientes do que nos cerca é essencial o conhecimento dessa disciplina; no entanto escolas e projetos devem sempre buscar

respeitar o crescimento cognitivo desses jovens e trabalhar sobre o propósito de interação de grupos e atividades lúdicas.

4 A PROGRAMAÇÃO NO NÍVEL SUPERIOR

Os cursos de engenharia ofertados nas universidades do país e do mundo estão diretamente ligados à área de tecnologia. Sendo assim, é comum observar em suas grades curriculares uma gama de disciplinas que utilizam *softwares* e composição de trabalhos usando linguagens de programação nos mais diferentes níveis e modelos.

Como forma de introduzir os alunos recém-ingressos nos cursos de engenharia, as coordenações tendem a compor no primeiro ano/semestre uma grade que forneça conhecimentos iniciais de programação e lógica matemática para esses alunos. Mas, devido à complexidade das disciplinas conseguintes, fica o questionamento sobre a suficiência dessa introdução e a eficácia que a mesma representa no desenvolvimento do futuro profissional.

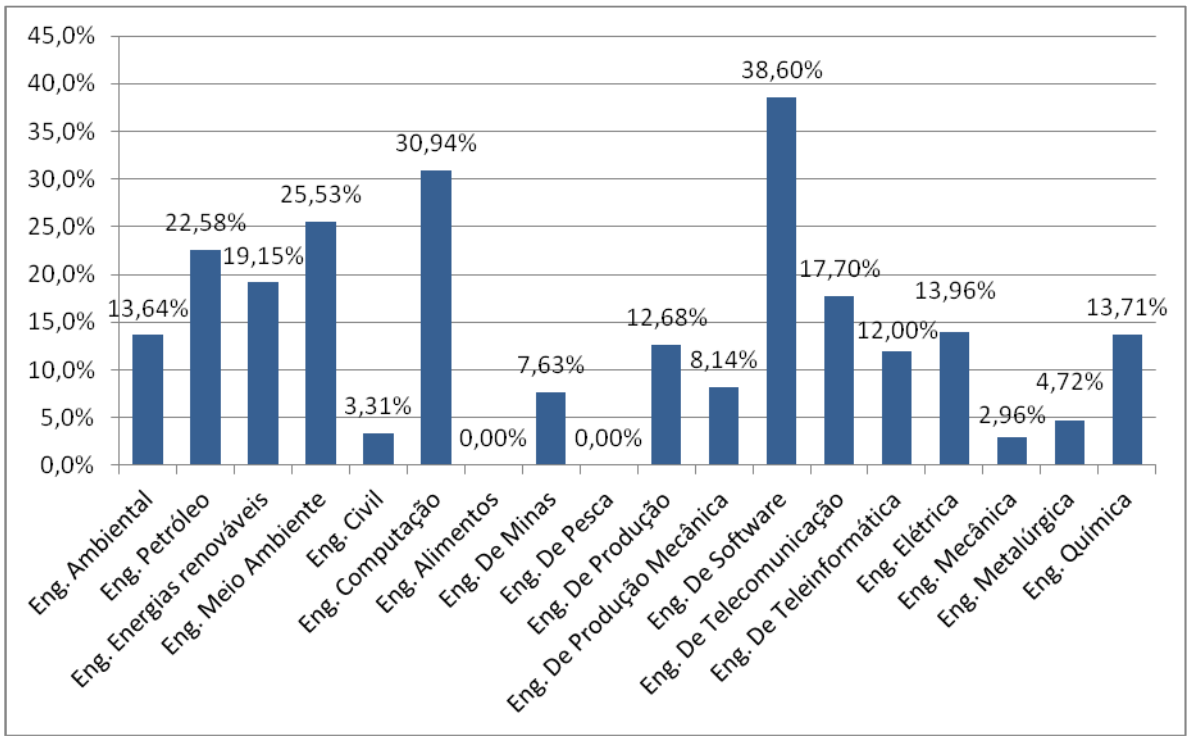
Esse capítulo trata então da presença da ciência da programação nos cursos de nível superior (em especial as engenharias) e o impacto que essa ciência causa nos anos iniciais do aluno ingresso na academia; cria ainda um posicionamento a respeito da transferência de responsabilidade do ensino introdutório desse conhecimento entre a universidade e a escola.

4.1 Programação na grade curricular das engenharias

Enganam-se aqueles que acreditam que a linguagem e a lógica de programação estão presentes apenas em graduações que diretamente levam o nome de computação. Não é apenas a ciência ou a engenharia da computação que carrega em sua grade curricular e na atuação profissional o uso de softwares e elaboração de algoritmos, mas sim grande parte das engenharias e possíveis outros cursos relacionados à ciência e pesquisa como licenciaturas e atividades administrativas e econômicas.

Analisando a grade curricular de cursos de engenharia oferecida pela Universidade Federal do Ceará (UFC), foi possível fazer um estudo da relevância que programação possui dentro desse setor acadêmico. O gráfico mostrado na Figura 7 representa a porcentagem da carga horária de disciplinas que utilizam diretamente programação em relação à carga horária total do curso. A Equação 1 mostra como foi realizado o procedimento para obtenção dos dados.

Figura 7 - Carga horária dedicada à programação em cursos de engenharia da UFC.



Fonte: Próprio autor (2017)

$$P(\%) = \frac{\sum C_{Dp}}{C_T - (C_{ES} + C_{TCC} + C_{HE})} \; x \; 100 \qquad (1)$$

Onde:

P = Porcentagem de carga horária de disciplinas que usam evidentemente programação em relação à carga horária total do curso.

CDP = Carga horária de disciplinas obrigatórias que usam programação.

CT = Carga horária total do curso em horas.

CES = Carga horária destinada ao estágio supervisionado em horas.

CTCC = Carga horária destinada ao trabalho de conclusão do curso em horas.

CHE = Carga horária destinada às atividades de horas extras.

O estudo foi feito sobre as disciplinas obrigatórias dos cursos, no entanto, dependendo das habilidades e interesses dos alunos, a porcentagem apresentada pode aumentar caso este usufrua de disciplinas optativas que se tenham afinidade com técnicas de programação.

É notório que, com exceção da Engenharia de Alimentos e a Engenharia de Pesca, todos os cursos de engenharia ofertados pela UFC possuem disciplinas que se relacionam com linguagem e lógica de programação e que algumas chegam a possuir uma carga horária superior a 10% como é o caso da Engenharia Elétrica, de Software, da Computação, Ambiental e Petróleo.

4.2 Programação na grade curricular da Engenharia Elétrica da UFC

Através dos dados obtidos pelo gráfico da Figura 7 é possível perceber a grande participação da lógica de programação na grade curricular do curso de Engenharia Elétrica ofertado pela Universidade Federal do Ceará, valor correspondente a 14% das disciplinas obrigatórias.

Assim como na grande maioria das engenharias, o primeiro ano do curso oferece como obrigatória a disciplina de “Programação computacional para engenharia” com uma carga horária de 96h. Durante essa disciplina, o aluno terá contato com os conceitos fundamentais de programação e elaborará pequenos projetos utilizando de uma linguagem de programação específica, normalmente a linguagem *C++*.

O aprendizado dessa disciplina no primeiro ano será refletido nos demais semestres de graduação. A seguir uma lista com conteúdos de caráter obrigatório que se utiliza de programação juntamente com a descrição da atividade relacionada a esta especialidade.

Eletrônica Digital: Nessa disciplina, o aluno compreende o uso de meios digitais para armazenar e manipular quantidades. É introduzido então o conceito de linguagem de máquina (binária) e portas lógicas e, de forma prática, alunos terão contato com o *software* PROTEUS, “[...] uma eficiente ferramenta computacional bastante utilizada em eletrônica digital, pois possui vastas bibliotecas com portas lógicas e ferramentas extras para simulação. Tem interface dinâmica, muitas funções e é fácil de utilizar.” (PET - ENGENHARIA ELÉTRICA UFC, 2014). Esse *software* cria circuitos integrados através de uma linguagem de programação com blocos gráficos.

Microprocessadores: Após a conclusão de Eletrônica Digital, o aluno é convidado a participar da disciplina de microprocessadores. Neste ambiente, além do conhecimento teórico a respeito dos microprocessadores, tem-se a utilização prática dos mesmos através de projetos que envolvem a programação em linguagem *assembly*. No final

do semestre, grupos de alunos devem desenvolver um projeto utilizando microprocessadores e a lógica de programação envolvida.

Métodos Numéricos Aplicados á Engenharia: Uma das ferramentas matemáticas mais utilizadas por engenheiros que nesse curso ganha uma versão computacional. Em laboratório o aluno utilizará do *software MatLab* para implementar os métodos desenvolvidos em sala de aula, ganhando dessa forma habilidades de análise de complexos modelos matemáticos provenientes da área de engenharia elétrica. (DEPARTAMENTO DE ENGENHARIA ELÉTRICA - UFC , 2005)

Sistemas Lineares: A matemática é utilizada como ferramenta para resolver problemas ligados à área de processamento de sinais digitais, sistemas de comunicação e sistemas de controle**. (**DEPARTAMENTO DE ENGENHARIA ELÉTRICA - UFC, 2005**).** Após a explanação teórica, utiliza-se novamente o *software MatLab* e a biblioteca *Simulink* para implementação dos sistemas e análise de entrada e saída de dados.

Controle de Sistemas Dinâmicos: Assim como em sistemas lineares, essa disciplina usa-se do *software MatLab* para programar sistemas de controle e representações matemáticas de sistemas elétricos. A biblioteca *simulink* é novamente utilizada para análise de raízes e polos de forma gráfica e também para projetar controladores PIDs (Controlador Proporcional Integral Derivativo).

Os tópicos acima citados são exemplos onde linguagens de programação são encontradas de forma direta dentro da carga horária obrigatória da Engenharia Elétrica da UFC. Caso seja de interesse do aluno, pode-se ainda aprofundar na área de robótica e eletrônica através da carga horária optativa, onde novamente, a programação estará presente de forma efetiva.

Nota-se, portanto, através desse estudo, a importância e a presença da ciência defendida nesse projeto dentro da grade curricular do curso de engenharia elétrica. No entanto, a seção seguinte irá mostrar que muitas vezes a disciplina de introdução à programação ofertada no primeiro ano não se torna suficiente para o acompanhamento no decorrer do curso, resultante de uma deficiência no ensino de lógica matemática e de programação no decorrer da educação básica.

4.3 Relação entre alunos de graduação e estudo de programação

Segundo Castro et. al (2003) é comum os currículos de cursos, como Ciência da Computação e engenharias, buscarem alternativas para diminuir seus índices de evasão; sendo

um dos grandes motivos, para os índices de evasão, a dificuldade de aprendizado. Aprender lógica de programação está entre as principais dificuldades destes alunos, levando-os a reprovação, diminuindo sua autoestima e consequentemente levando à evasão (FERREIRA et. al, 2010).

A fim de averiguar estas proposições, desenvolveu-se um questionário online direcionado principalmente para alunos da UFC. O questionário teve como principal objetivo verificar a familiaridade dos alunos de engenharia, ciência e tecnologia com a ciência da programação antes e durante a universidade; além disso, buscou a opinião dos entrevistados a respeito do ensino de programação durante o ensino básico.

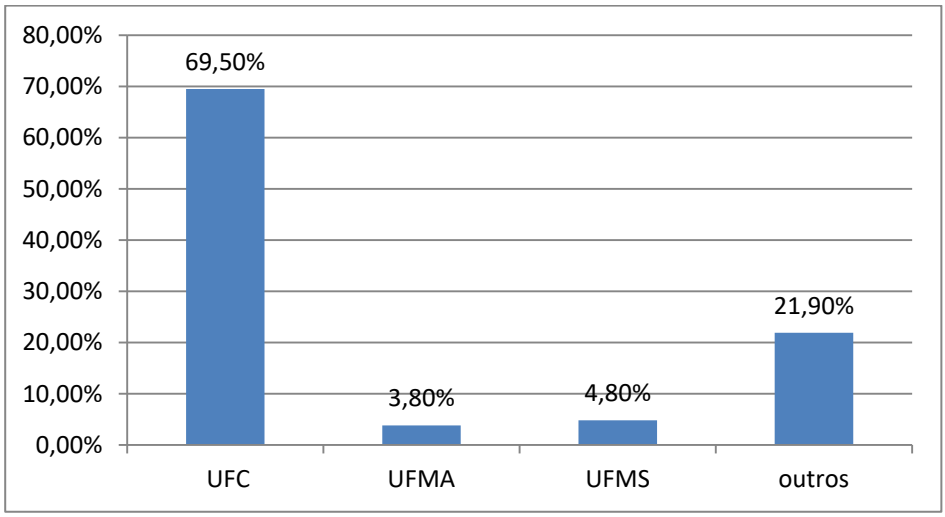
Os dados obtidos com o questionário estão descritos na subseção 4.3.1 e analisados na 4.3.2.

4.3.1 *Resultados obtidos*

O questionário intitulado como “Programação é para todos?” encontra-se no apêndice D e foi realizado entre os dias 25 e 27 de outubro de 2017, contando com a participação de 106 alunos de diferentes universidades e cursos.

O foco da pesquisa foi a Universidade Federal do Ceará que contou com aproximadamente 70,0% das respostas, sendo o formulário ainda respondido por outras universidades públicas e privadas do país (Figura 8). Dentre os cursos participantes da pesquisa, o modal se deu por alunos do curso de Engenharia Elétrica (cerca de 10,0% das respostas), seguido por 8,0% do curso de Computação.

Figura 8 - Participação das universidades na pesquisa.

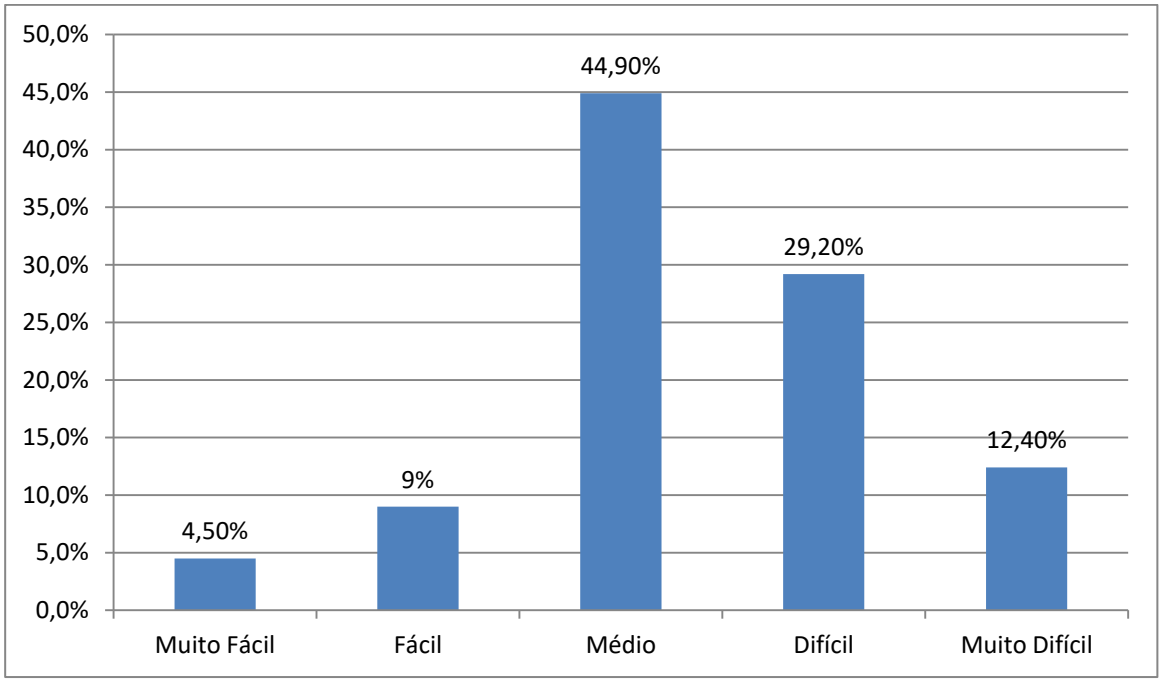


Fonte: Próprio autor (2017).

Dentre os entrevistados apenas 23,6% não obtiveram contato com programação durante a universidade. Estes alunos são oriundos de cursos da área de administração, licenciaturas e artes. Os outros 76,4% são, na maioria, alunos dos cursos de tecnologia, ciências da computação e economia.

Dentre os alunos que tiveram contato com programação durante a universidade, 41,6% consideram as disciplinas que envolvem essa ciência como difíceis ou muito difíceis, sendo que cerca de 45,0% avaliam como dificuldade média com relação as demais disciplinas da grade curricular dos seus cursos, conforme figura abaixo.

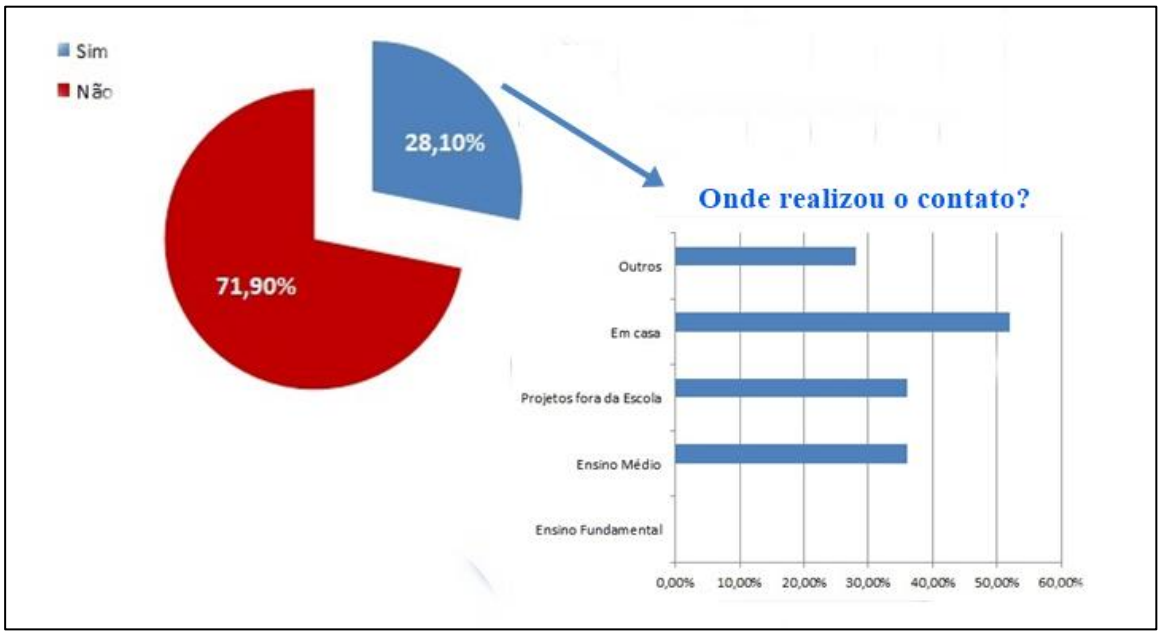
Figura 9 - Grau de dificuldade nas disciplinas de programação.



Fonte: Próprio autor (2017)

Dentre o grupo dos alunos que usaram programação dentro da universidade, apenas 28,10% já haviam tido contato com esta disciplina anteriormente ao ingresso no curso. A principal forma de contato foi através de métodos autodidatas dentro da própria casa (52,0%); sendo o ensino fundamental o portador do menor índice de inserção dessa ciência na educação dos jovens. Projetos fora da escola e atividades no ensino médio corresponderam a aproximadamente 35,0% cada. A Figura 10 mostra esses dados.

Figura 10 - Contato com programação anteriormente à universidade.



Fonte: próprio autor (2017).

Ao final do questionário, foi solicitada a opinião dos participantes a respeito da importância da inserção de lógica e linguagem de programação antes do ingresso na universidade (durante a educação básica). Dos 106 participantes, 95,5% acreditam que é sim importante essa inserção; isso inclui mesmo aqueles que não tiveram contato com essa ciência dentro da universidade.

4.3.2 *Considerações*

A pesquisa realizada, contando com 106 alunos de diferentes universidades do país, demonstrou algumas das proposições apresentadas nesse trabalho.

É possível perceber através da Figura 9 que 41,6% dos participantes sentem dificuldade em disciplinas que utilizam de programação, reafirmando a teoria de que o curso introdutório dessa disciplina no primeiro ano na universidade é, em grande parte, insuficiente para a continuação do entendimento de lógica de programação no decorrer da vida acadêmica.

Dentro do grupo focal, apenas 28,10% tiveram algum contato com linguagem de programação antes de iniciarem a carreira no curso de ciência e tecnologia. A maior deficiência encontrada foi durante o período de ensino fundamental, sendo que nenhum dos entrevistados teve contato durante esse período escolar. Ficou claro que, dos que tiveram esse contato, grande parte teve essa inserção de forma autodidata, sendo realizada dentro da

própria casa dos entrevistados. É perceptível a deficiência ou a inexistência da oferta desse ensino pelas escolas de educação básica.

Por último, é notória a percepção dos alunos das áreas de tecnologia e ciência de que o ensino de lógica e linguagem de programação é de grande importância durante o período infantil/juvenil.

5 ESTUDO DE CASO: CURSO INTRODUTÓRIO À LINGUAGEM DE PROGRAMAÇÃO - *PYTHON GAMES*.

Após a realização do estudo teórico e da pesquisa entre alunos da universidade, dispostos nos capítulos 3 e 4 respectivamente, percebeu-se a importância da prática de programação durante o ensino básico dos alunos e a dificuldade que os graduandos adquirem em disciplinas que envolvem lógica de programação durante o curso.

Sendo assim, buscando envolver as metodologias estudadas no capítulo 3, idealizou-se aplicar uma nova metodologia de ensino de programação para alunos do ensino médio das escolas públicas da cidade de Fortaleza – CE através da linguagem de programação *Python*.

Os principais objetivos desse estudo de caso são:

- Desenvolver uma metodologia que proporcione um melhor aprendizado de lógica de programação em alunos da rede básica, em especial alunos do ensino médio;
- Verificar a capacidade de aprendizado de alunos pertencentes ao ensino médio em relação à ciência programação, em especial analisar a capacidade de se desenvolver lógica de programação nessa classe de alunos;
- Observar o grau de interesse de alunos por essa disciplina e examinar a opinião da classe estudada sobre a importância de conhecer essa ciência durante o ensino básico.

As seções desse capítulo, que se seguem, trarão com detalhes o desenvolvimento do projeto, o conteúdo programado e a metodologia aplicada. Os resultados obtidos serão discutidos no final do capítulo.

5.1 Idealização do projeto

O projeto de estudo de caso foi pensado durante a estadia do autor na bolsa de extensão oferecida pela PREX (Pró-Reitoria de Extensão da UFC) realizado no espaço Seara da Ciência pertencente à Universidade Federal do Ceará durante o período de Fevereiro à Dezembro de 2016.

A Seara da Ciência visa à propagação científica e tecnológica à sociedade. Localizada no cruzamento das ruas Dr. Abdênago Rocha Lima com a Av. Andrade Furtado, no Campus do Pici, oferece uma série de atividades e projetos voltados, principalmente, para

alunos da rede pública da cidade de Fortaleza e do interior do Ceará. Dentre as atividades, existem os cursos básicos com os seguintes temas: Matemática, Química, Biologia, Física e Astronomia, além do salão de exposição com experimentos, práticas pedagógicas e educativas diversas e do salão de teatro, onde alunos desenvolvem projetos teatrais e exposição de trabalhos científicos.

Além dos espaços mencionados, a Seara da Ciência conta também com um laboratório de informática onde normalmente se realizam trabalhos complementares aos cursos básicos, levando o aluno a realizar pesquisas diversas e usar *softwares* para estudar astronomia. O laboratório serve ainda como sala de estudo para os bolsistas que usam os momentos livres para fazer pesquisas e realizar trabalhos. Mesmo com essas práticas, percebeu-se que o tempo de exposição do laboratório de informática para projetos voltados à sociedade era insuficiente, levando o autor a idealizar o projeto de inserção de programação para alunos da rede básica e realizar o estudo de caso proposto pelo capítulo 5.

5.1.1 Público alvo

O público alvo escolhido foram alunos do ensino médio, de preferência pertencentes à escola pública da cidade de Fortaleza-CE, cidade sede da instituição onde o curso foi desenvolvido. Para a tomada dessa escolha, levaram-se em consideração três fatores principais:

1. O projeto foi idealizado para ser desenvolvido no espaço da Seara da Ciência, local pertencente à Universidade Federal do Ceará. Visto isso, preferiu-se por evitar trabalhar com alunos do ensino fundamental, pois estes provavelmente precisariam de acompanhamento dos pais até o local. Alunos de ensino médio possuem uma maior autonomia em relação ao deslocamento.
2. Durante as inscrições para os cursos básicos[3] a Seara da Ciência guarda uma gama de informações a respeito de alunos do ensino médio (público alvo do projeto). Como as inscrições do projeto a ser realizado foram feitas fora do período de inscrições para os cursos básicos, utilizou-se esse banco de dados para convidar alunos a participarem desse trabalho.

[3] Os cursos básicos é um projeto realizado pela Seara da Ciência que visa oferecer aos alunos do ensino médio das escolas públicas de Fortaleza-CE uma revisão do conteúdo básico de disciplinas como matemática, física, química, biologia e astronomia.

3. Ainda, sendo essa a primeira experiência do autor em lecionar programação, preferiu-se por uma faixa etária de alunos mais elevada, sendo estes mais independentes e com capacidade educacional para melhor atender às proposições do projeto e responder aos questionários de satisfação.

5.1.2 Linguagem de programação e tema a ser desenvolvido

Através do estudo teórico feito nos capítulos 2 e 3, foi possível adquirir noções de projetos já realizados anteriormente por outros pesquisadores, professores e estudantes e assim definir uma estratégia pedagógica para realizar o trabalho.

Ficou claro que a linguagem *Scratch* é a mais utilizável em cursos onde o objetivo é levar a lógica de programação pela primeira vez aos alunos. No entanto, observou-se que pesquisadores que defendem essa linguagem, aplicaram-na em projetos com alunos do ensino fundamental devido a sua simplicidade e facilidades gráficas. Como o projeto aqui estudado direciona-se para alunos do ensino médio, optou-se por escolher uma linguagem de alto-nível, mas que escreva algoritmos em forma de texto e não em forma de blocos e desenhos. Escolheu-se então a linguagem *Python* baseado nos argumentos defendidos na seção 2.3 desse trabalho.

Observou-se ainda, conforme a seção 2.3.4 deste trabalho, que a proximidade de alunos com a programação se dá quando estes utilizam esse recurso para desenvolver trabalhos que envolvam jogos, desenhos, animações. Sendo assim, optou-se por trabalhar a linguagem *Python* com o uso de uma biblioteca[4] denominada *Pygame*. A biblioteca *Pygame* é de fácil acesso e contém uma gama de comandos que facilitam a criação de imagem, animação e por consequência a criação de jogos.

Devido a essas escolhas, intitulou-se o projeto como: *Curso Introdutório à Linguagem de Programação – Python Games*. A Figura *11* mostra o *slogan* de divulgação do projeto que contém suas principais informações direcionadas aos alunos.

[4] Biblioteca em programação diz respeito a um conjunto de comandos compactados e transformados em funções que podem ser usadas em outros programas.

Figura 11 - Slogan utilizado para divulgação do projeto.

Fonte: Próprio autor (2016).

5.2 Concepção do projeto

Para realização do projeto, verificou-se a qualidade dos computadores presentes no laboratório de informática da Seara da Ciência e a compatibilidade com os programas necessários para desenvolver os trabalhos, obtendo a quantidade de 15 máquinas prontas para o uso. Para evitar superlotação da turma e um melhor desenvolvimento dos estudantes, foi decidida a abertura de uma turma de também 15 alunos, dessa forma cada aluno poderia trabalhar e executar as atividades em um computador próprio.

Como material de apoio para as aulas, o autor juntamente com a coordenação da Seara da Ciência, optou pelo uso do equipamento projetor e pela criação de uma apostila que foi disponibilizada de forma impressa e digital. O autor desse trabalho ficou também encarregado de ser o monitor facilitador durante o curso, direcionando as aulas e esclarecendo dúvidas dos alunos. Para uma melhor comunicação entre monitor e aluno fora dos horários de aula, criaram-se também um grupo na rede social *Whatsapp* e no site de compartilhamento *Dropbox*.

As subseções a seguir relatam a concepção do projeto, desde as inscrições dos alunos, como o planejamento das aulas e a criação da apostila de apoio.

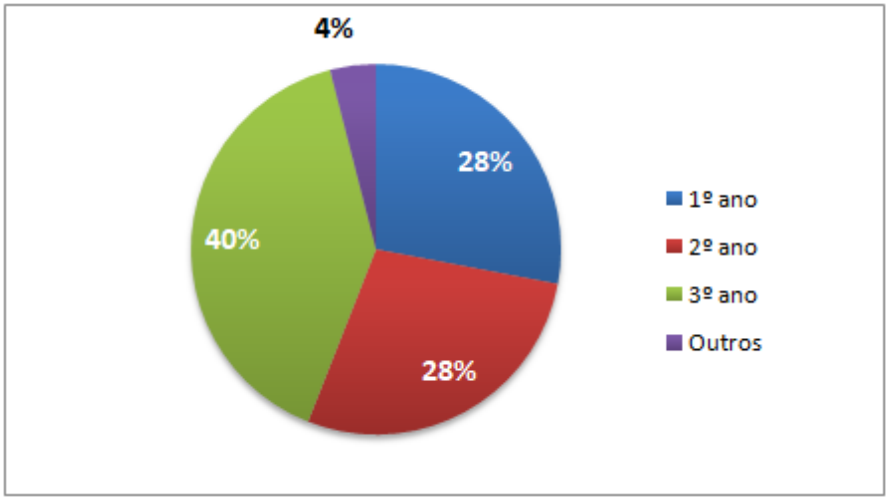
5.2.1 Inscrições e perfil dos alunos

As inscrições para o projeto ocorreram durante os dias 6 e 11 de setembro de 2016, através do formulário *online* do *Google Docs* disposto no apêndice A desse trabalho. Utilizou-se o sítio da Seara da Ciência para realizar a divulgação, além da fixação de cartazes nos flanelógrafos internos ao ambiente. Para aumentar o alcance da divulgação, usou-se ainda de emails de alunos pertencentes ao banco de dados da Seara da Ciência.

Como forma de seleção dos inscritos, foram feitas, além dos dados básicos dos estudantes, perguntas que indicassem o grau de interesse pela disciplina de matemática, por computadores e por jogos, e uma pequena descrição sobre qual a expectativa de aprendizado no curso que estava a se inscrever.

Houve um total de 22 inscritos, sendo selecionado os 15 que melhor mostraram interesse no ato da inscrição online. A escolha foi feita pelo autor usando as informações oferecidas pelos alunos no formulário de inscrição. Teve-se uma composição de 96% de inscritos oriundos da escola pública, sendo 40% do total pertencentes ao 3º ano do ensino médio, 28% do 2º ano e outros 28% pertencentes ao 1º ano (Figura 12).

Figura 12 - Faixa escolar dos alunos inscritos no projeto.



Fonte: Próprio autor (2016).

Ainda segundo o formulário, dos alunos inscritos 56% nunca tiveram contato com programação. Os demais alunos realizaram atividades, com essa temática, fora da escola em cursos anteriormente ofertados pela Seara da Ciência, o que induz ao pensamento de que provavelmente as escolas de nível básico da cidade de Fortaleza/CE não ofertam esse tipo de conhecimento aos seus alunos.

5.2.2 Pré-elaboração do projeto

Como dito na seção 5.1.1 desse trabalho, este projeto foi a primeira experiência do autor com a prática de lecionar a ciência de programação para alunos da rede básica. Sendo assim, a didática de ensino e as escolhas tomadas para concepção desse trabalho foram

reflexos do estudo bibliográfico realizado nos capítulos anteriores e da experiência do autor no ato de lecionar outras disciplinas.

O autor, graduando o curso de engenharia elétrica da UFC, possui grau de conhecimento de programação principalmente em linguagens *Assembly*, *C++* e *VHDL*. Já o conhecimento de linguagem *Python* foi adquirido através do curso online intitulado "*Program Arcade Games with Python and Pygame*" disponibilizado no site <http://www.programarcadegames.com>.

As disposições das aulas mostradas na seção 5.2.4 foram planejadas em concordância com as aulas do curso online, sendo algumas delas reduzidas para tornar o curso menos extenso, mas com conteúdo suficiente para se completá-lo.

5.2.3 Elaboração do material de apoio

As aulas e o conteúdo estruturado do curso foram baseados em um projeto americano desenvolvido pelo professor Dr. Paulo Vicent Craven, disponível online, intitulado "Programa de Jogos Arcade utilizando *Python* e *Pygame*[5]"(Tradução Livre) que conta com 20 capítulos de instruções e exercícios em linguagem *Python* destinado para alunos da rede básica. O projeto tem como objetivo facilitar e desenvolver o raciocínio lógico e técnicas de programação através da criação de jogos do estilo arcade.

O pesquisador desenvolveu então uma apostila com 11 capítulos que mimetizam o projeto online. Cada capítulo apresenta uma explanação teórica sobre a utilização de comandos e recursos, além de um conjunto de exercícios teóricos e práticos, onde o aluno tende a assimilar a lógica matemática e as características de escrita da linguagem *Python*. Ao final de cada capítulo existe uma seção intitulada "projetando" onde é proposto um determinado problema e o aluno elabora no computador uma resolução.

A seguir um resumo e explanação sobre cada capítulo. A apostila está disposta no apêndice E para consultas.

Capítulo 1 (Introdução): Neste capítulo é discutida com os alunos a justificativa e o objetivo do projeto além de relatar a história e a evolução das linguagens de programação. Utilizando-se desse momento o monitor deve introduzir os conceitos da linguagem *Python* e do IDLE utilizado, mostrando ainda como baixar os aplicativos de forma gratuita e sua instalação no computador.

[5] Título original Program Arcade Games with Python and Pygame. Para informações a respeito deve-se consultar o endereço eletrônico do projeto disponível no sítio apresentado nas referências bibliográficas.

Capítulo 2 (Algoritmo): É inserido nesse capítulo o conceito de ‘Algoritmo’. O aluno é levado a criar algoritmos de atividades simples como cozinhar ou tomar banho, e escrevê-los de forma detalhada e com uma sequência lógica de passos. O monitor deve aproveitar o momento para inserir atividades lúdicas e dinâmicas, evitando focar apenas em algoritmos de forma escrita.

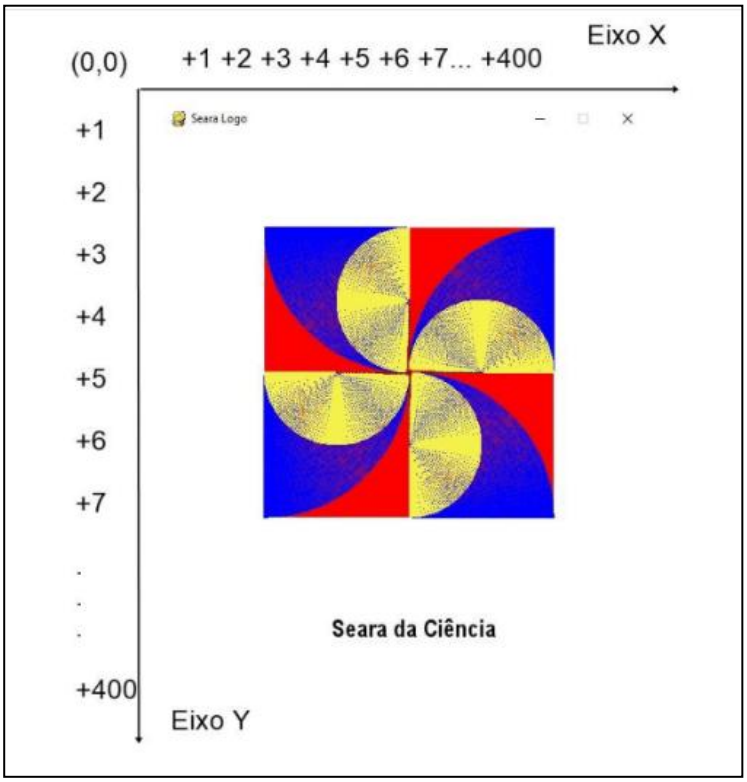
Capítulo 3 (Hello Word): Nesse capítulo são inseridos os primeiros comandos e as primeiras atividades ligadas diretamente ao computador. O comando *print()* e o comando *float()*, usados para enviar e receber informações entre computador e usuário, são exemplificados e conectados com expressões e operações matemáticas. Ao final do capítulo o aluno é desafiado a criar um programa que resolva equações básicas. Como sugestões têm-se o cálculo da velocidade média, a média de notas de um aluno e a renda mensal de uma família.

Capítulo 4 (Condições): Aqui é introduzido o comando *if()* e seus auxiliares *else* e *elif()*, chamadas de funções condicionais. Essas funções estão entre as mais importantes no curso que irá se seguir, sendo então de grande importância a verificação do monitor perante os alunos participantes se há dúvidas ao final do capítulo. Como projeto proposto, o capítulo traz a elaboração de um *quis*, onde os alunos devem desenvolver um conjunto de perguntas e respostas e oferecer a outro estudante para responder às questões. O programa criado deve informar quando a resposta for correta ou errada.

Capítulo 5 (Repetições): Juntamente com os comandos condicionais, os comandos de repetição serão de grande importância no desenvolver do projeto. Aqui o capítulo introduz o conceito e exemplifica a gramática para se operar com os comando *for()* e *while()*. O principal objetivo do capítulo é oferecer condições para o aluno diferenciar as duas estruturas e perceber quando devem ser utilizadas. Ao final do capítulo os alunos são convidados a criar o primeiro jogo (sem gráficos, apenas com escritas e opções que descrevem a escolha da personagem do jogo).

Capítulo 6 (Introdução aos gráficos): Tendo sido fixados os principais comandos em *Python*, o capítulo 6 traz como proposta a criação das primeiras imagens. Aqui são introduzidos os conceitos de cores RGB (*Red*, *Green*, *Blue*), plano cartesiano aplicado à linguagem *Python* e comandos que facilitam a criação de formas geométricas na tela. A Figura 13 a seguir mostra o desenho base para imersão desse tema de acordo com a apostila. Ao final do capítulo propõe-se ao aluno escrever um programa que desenhe uma paisagem ou algo de sua preferência que contenha diferentes cores e formatos.

Figura 13 – Figura base para discussão do capítulo 6.



Fonte: Próprio autor (2016).

Capítulo 7 (Listas e tuplas): ‘Lista’ é uma variável que pode guardar uma quantidade de elementos definida pelo programador, podendo ser modificadas ao longo do programa. A ‘tupla’ não pode ser modificada, operando com a mesma quantidade de elementos durante todo o processo. Grande parte dos jogos trabalha com essa ferramenta, onde podem guardar informações como porcentagem de vida, itens, experiência de uma personagem. Ao final do capítulo o aluno criará seu segundo jogo, novamente sem utilizar informações gráficas, mas guardando informações a respeito da personagem (usuário) que irá jogar.

Capítulo 8 (Introdução à animação): Esse capítulo traz uma junção dos capítulos 6 e 5, onde através de repetições uma determinada imagem pode parecer uma animação. Na seção ‘projetando’ o aluno é convidado a criar uma tela onde simule um campo e um nevoeiro. Para isso, o capítulo traz um programa já construído onde o aluno precisa compreender e fazer modificações necessárias para que o nevoeiro aumente a velocidade de queda dos flocos, ou para realizar mudanças nas cores de fundo ou dos flocos de neve.

Capítulo 9 (Funções): As funções têm como objetivo simplificar e enxugar os programas criados. Dessa forma, ao invés de escrever repetidas vezes um mesmo conjunto de

dados, pode-se transformá-los em uma função com parâmetros de entrada e saída, dessa forma pode-se usar a função em diversas partes do programa. Na seção ‘projetando’ o aluno tem como objetivo criar cinco funções distintas, cada uma buscando utilizar comandos definidos em capítulos anteriores.

Capítulo 10 (Controles): Nesse capítulo o aluno tomará posse de comandos que visam o controle, através do teclado e do *mouse*, de animações e tela. Com esses conhecimentos é então possível criar jogos simples e funcionais. Como atividade proposta o aluno deve criar uma personagem gráfica simples e fazê-la movimentar-se pela tela usando comandos de teclado. Abre-se ainda um momento para incentivar o aluno a permanecer treinando as habilidades e a buscar novas dicas em outros materiais de estudo para que se possam realizar projetos maiores e com capacidades de gráficos melhor elaborados.

Capítulo 11 (Lista de comandos principais): No último capítulo tem-se uma lista com os comandos trabalhados durante os outros capítulos do curso, seguidos de suas utilidades.

Ficou acordado entre o autor e a coordenação da Seara da Ciência que a apostila desenvolvida seria disponibilizada para os alunos em forma digital (PDF) e ainda de forma impressa, ficando a responsabilidade da confecção por parte da coordenação.

5.2.4 Carga horária e finalização

O projeto foi organizado de forma a ter inicio no dia 13 de setembro de 2016 e com previsão de término para 5 de outubro de 2016, havendo três aulas por semana (nos dias de terça-feira, quarta-feira e quinta-feira) no horário de 14:00 às 17:00 horas. Dessa forma, totalizando carga horária de 30 horas com o planejamento das aulas disposto conforme Tabela 2.

Tabela 2 - Planejamento das aulas.

Dia	*Conteúdo Programado*
13/09/2016	*Recepção dos alunos. Levar os alunos a conhecer os espaços da Seara da Ciência. Associar experimentos dispostos no salão com o uso de computadores. Trabalhar o capítulo 1 da apostila fora da sala de informática. Levar os alunos a conhecerem o laboratório. Esclarecer os objetivos do projeto e dúvidas a respeito das*

	atividades. Realizar capítulo 2 da apostila.
14/09/2016	*Iniciar capítulo 3 e realizar entrega da apostila física. Iniciar capítulo 4.*
15/09/2016	*Desenvolver o capítulo 4 e acompanhar os alunos no desenvolvimento da seção 'projetando'.*
20/09/2016	*Desenvolver o capítulo 5 e acompanhar os alunos no desenvolvimento da seção 'projetando'*
21/09/2016	*Desenvolver o capítulo 6 e acompanhar os alunos no desenvolvimento da seção 'projetando'*
22/09/2016	*Não haverá aula.*
27/09/2016	*Desenvolver o capítulo 7 e acompanhar os alunos no desenvolvimento da seção 'projetando'*
28/09/2016	*Desenvolver o capítulo 8 e acompanhar os alunos no desenvolvimento da seção 'projetando'*
29/09/2016	*Desenvolver o capítulo 9 e acompanhar os alunos no desenvolvimento da seção 'projetando'*
04/10/2016	*Desenvolver o capítulo 10 e acompanhar os alunos no desenvolvimento da seção 'projetando'. Orientar os alunos a respeito da importância do último dia onde serão aplicados os testes e a entrega de certificado*
05/10/2016	*Realização de dois testes escritos com os alunos. O primeiro com caráter de consulta de opinião a respeito das atividades ministradas e sobre as condições físicas oferecidas pelo curso. O segundo teste com o intuito de averiguar os conhecimentos adquiridos pelos alunos ao longo dos dez encontros. Entrega de certificados de participação.*

Fonte: Próprio autor (2016).

Ficou ainda acertada com a coordenação da Seara da Ciência a emissão de certificado para os alunos participantes, como comprovação do cumprimento das 30 horas de aula para os alunos que cumprissem ao menos 75% da frequência. O modelo do certificado é mostrado na Figura 14 abaixo.

Figura 14 - Modelo do certificado emitido para os participantes do curso.

Fonte: Próprio Autor (2016).

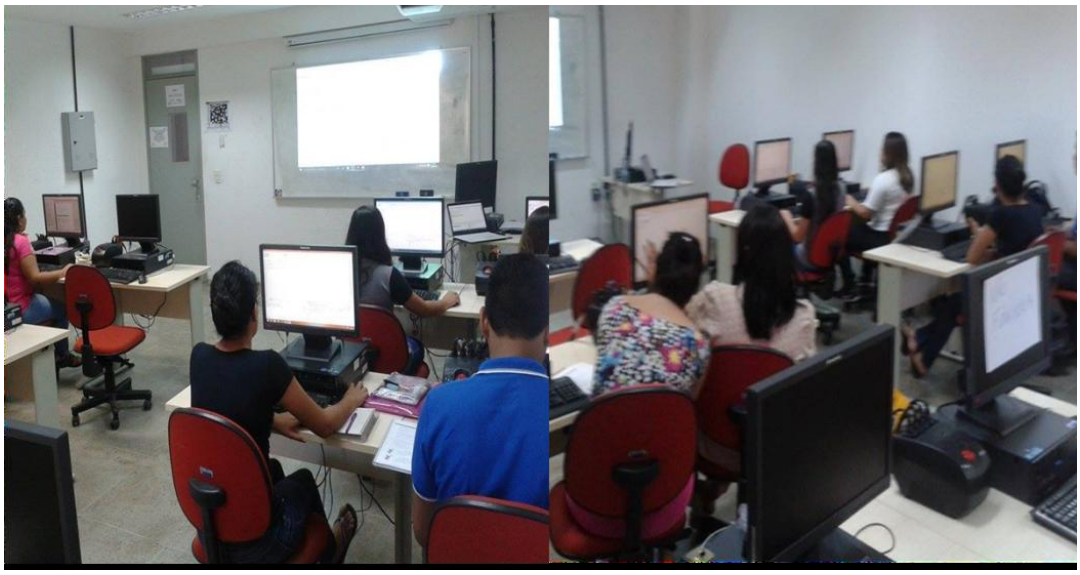
5.3 Desenvolvimento do projeto

O desenvolvimento do projeto pode ser resumido em alguns pontos importantes:

- O projeto foi realizado conforme cronograma disposto na Tabela 2 sem mudanças expressivas.
- Alguns computadores ofereceram dificuldades em realizar determinadas funções e comandos da linguagem *Python*. O problema não foi identificado, levando alguns alunos a programar em dupla nas primeiras três aulas.
- O capítulo 10 da apostila ofereceu o maior grau de dificuldades entre os alunos. Devido ao tempo, não foi possível explanar todas as dúvidas provenientes da seção ‘projetando’ do capítulo.

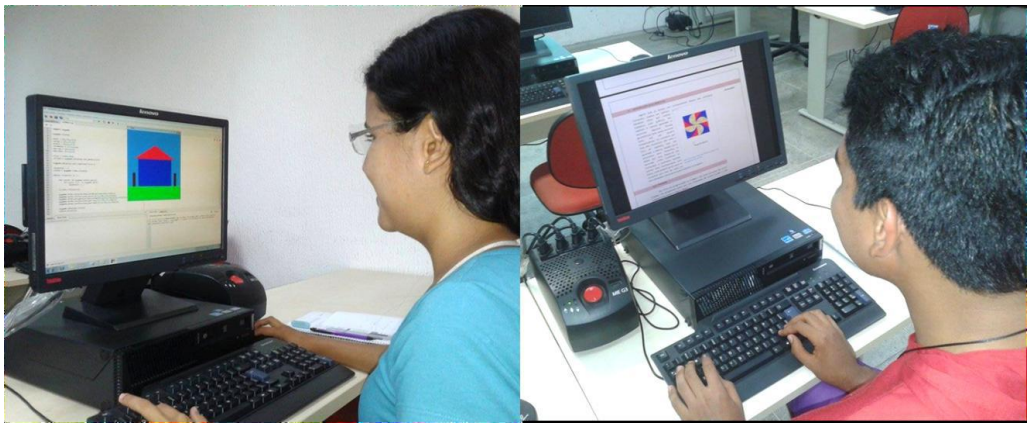
A Figura 15 mostra a disposição dos alunos pelo espaço do laboratório de informática, enquanto que a Figura 16 mostra os alunos trabalhando nos projetos propostos.

Figura 15 - Disposição dos alunos na sala de aula.



Fonte: Próprio autor (2016).

Figura 16 - Alunos desenvolvendo projetos sugeridos.



Fonte: Próprio autor (2016).

5.4 Resultados obtidos

O curso teve início conforme o previsto, no dia 13 de outubro de 2016, com a participação de 15 alunos. No entanto, ocorreu uma desistência de 33% desses alunos, tendo o curso finalizado com 10 alunos. Contatando os desistentes, percebeu-se que o motivo principal estava relacionado ao não atendimento das expectativas a respeito do que seria o trabalho proposto.

Como forma de avaliar o impacto que o curso de introdução à programação teve no cenário acadêmico dos alunos participantes, optou-se por aplicar dois testes no ultimo dia (05 de outubro de 2016). Os testes foram realizados de forma que o aluno não precisava se identificar e podem ser consultados nos apêndices B e C, contando com as seguintes características.

Teste 1 - Avaliação do curso: esse teste teve como objetivo averiguar os motivos que geraram dificuldades para o aluno durante todo o curso e a satisfação destes com o que foi desenvolvido em sala. Foram realizadas nove perguntas objetivas, que indicavam o índice de satisfação do aluno com os fatores a serem analisados. A Tabela 3 relaciona os itens analisados com a porcentagem de alunos que marcaram as opções “pouco”, “médio” e “muito” como índice de satisfação.

Tabela 3 - Resultado referente ao teste 1.

Item averiguado	Grau de satisfação (%)		
	Pouco	Médio	Muito
Metodologia do curso (uso de apostila, *data show*, computadores, *dropbox* e *whatsapp*)	0	0	100
Escolha da linguagem de programação utilizada (Python).	10	20	70
Didática do monitor.	0	20	80
Estrutura da sala de aula (Laboratório de Informática da Seara da Ciência).	20	30	50
Horários e dias da semana escolhidos para realização das atividades (Terça/Quarta/Quinta – 14:00 às 17:00).	0	0	100
O curso mudou sua forma de ver o mundo.	0	30	70
Desejo de seguir na área de programação (faculdade, curso técnico etc.).	40	40	20
Chances de recomendar o curso para outros amigos.	0	10	90
Grau de satisfação com o curso.	0	20	80

Fonte: Próprio autor

Através dos dados obtidos na Tabela 3 é notório que os alunos tiveram problemas com os computadores oferecidos pelo laboratório de informática da Seara da Ciência (50% dos alunos). E ainda cerca de 30% dos remanescentes tiveram dificuldades com a linguagem de programação *Python*, não estando estes de total satisfação com a linguagem escolhida.

No entanto, essas dificuldades não pareceram atrapalhar o grau de satisfação dos alunos para com o curso ministrado. O grau de satisfação foi de 80% com alunos muito satisfeitos, sendo que 90% recomendaria o curso para amigos.

Teste 2 – Avaliação de assimilação de conteúdo pelo aluno: O segundo teste teve como objetivo averiguar o grau de absorção que os alunos tiveram em relação à lógica de programação e conceitos teóricos e práticos da linguagem *Python* e da biblioteca *Pygame.* O teste contava com três questões: a primeira de caráter teórico, a segunda utilizando os comandos básicos desenvolvidos até o capítulo 5 da apostila e a terceira sendo o desenvolvimento de um programa utilizando os comandos e a biblioteca *Pygame* relacionados com os capítulos 6-10 da apostila.

Estipulou-se que seria considerado como satisfatório, o nível de aprendizado, caso o aluno conseguisse responder às três questões, mesmo que com ajuda do monitor. Ao final do teste, foi percebível um índice satisfatório de 70% dos alunos, sendo os outros 30% considerados com índice mediano de absorção.

5.5 Considerações

Com o estudo de caso desenvolvido é possível responder alguns dos questionamentos suscitados durante este projeto e analisar possíveis melhorias e táticas para um novo desenvolvimento.

O ensino de programação para alunos do ensino médio se tornou eficaz dentro do campo amostral em que foi trabalhado (um total de dez alunos remanescentes no projeto), 70% desses alunos obtiveram resultados positivos no ‘Teste 2’ e apresentaram um crescimento positivo no que se diz respeito ao entendimento de lógica e sintaxe de programação, fatores analisados pelo monitor-facilitador do projeto.

Os alunos se sentiram entusiasmados ao fim do projeto como a nova visão de mundo adquirida e com os conhecimentos atingidos. Sentiram-se ainda confortáveis com a metodologia de ensino e o material didático apresentado.

Como problemas, detectou-se o mau funcionamento de alguns computadores.. Esse problema é de fácil solução, sendo os computadores acessíveis para formatação de *software* e com uma melhor análise podem vir a ser utilizados de forma correta e com funcionalidade para o projeto.

A linguagem de programação *Python* foi ainda analisada pelos alunos como dificultosa; para isso, pode-se ainda realizar a introdução da programação com linguagens

mais gráficas como *Scratch* ou *VisuAlg.* A linguagem *Python* pode ser então utilizada em um segundo momento, visando alunos da rede básica que tiveram anteriormente algum contato com programação.

O nível de desistência (33%) foi considerado alto. Entre estes alunos, dois justificaram a ausência por motivos de não terem encontrado o que esperavam. Os demais não enviaram notas justificando a desistência. Uma forma de minimizar esses acontecimentos em futuros projetos seria a divulgação do evento nas próprias escolas, de forma que, o divulgador possa explicar e sumarizar os conteúdos e metodologias do curso, para que alunos que não se identifiquem com o projeto evitem se inscrever.

6. CONCLUSÕES

Partindo do objetivo principal desse trabalho, foi possível a realização de uma pesquisa fundamentada a respeito do ensino de lógica de programação durante o ensino básico, concluindo que a inserção atual no mundo digital torna necessário o conhecimento dessa ciência durante essa fase da vida de forma a contribuir com o crescimento cognitivo do aluno e da sua capacidade de solucionar problemas de forma criativa e lógica.

Percebeu-se ainda através da pesquisa realizada com alunos de diferentes universidades que o ensino de programação durante a educação básica é sem dúvida um diferencial para aqueles que visam cursos universitários do ramo de tecnologia, sendo essa prática uma possível mitigadora de dificuldades e aversões que alunos universitários adquirem durante sua estadia no nível superior de ensino.

Para a obtenção de dados a respeito do desempenho de alunos de ensino médio no aprendizado de lógica computacional, o estudo de caso ofereceu algumas informações importantes para a fundamentação das proposições acima citadas. Primeiro que dentre os alunos participantes, 70% demonstraram um grau de absorção de conhecimento considerado satisfatório de acordo com o teste realizado pelo autor. Ainda notou-se que 80% dos participantes classificaram o curso como muito satisfatório, não obtendo valores significantes de descontentamento em relação ao material utilizado, à metodologia empregada e à linguagem de programação escolhida.

Por fim, desenvolveu-se uma metodologia de ensino de lógica de programação considerada satisfatória dentre os alunos participantes do estudo de caso. A metodologia usou-se da linguagem *Python* e da biblioteca *Pygame* através da temática *Arcade Games* para estimular o processo de criatividade dos alunos no aprendizado dessa ciência e fornecer um método de ensino que estimule o desejo do aluno em ampliar suas habilidades nessa área.

O trabalho teve sua importância na abordagem de temas voltados à área de educação levando-os a serem discutidos no ambiente de engenharia e trazendo a proposta de se trabalhar questões voltadas à prática de ensino dentro dos cursos de tecnologia da Universidade Federal do Ceará.

Fica como proposta de trabalhos futuros a aplicação de práticas de programação e robótica para alunos da rede básica, levando esses conhecimentos para o âmbito fora do grau técnico e superior. Cria-se ainda espaço para averiguação do impacto que essas práticas de ensino podem causar nesses alunos e o diferencial para a escolha e realização do curso de universitário.

REFERÊNCIAS

AMARAL, R. **Qual o melhor IDE para o python,** 2009. Disponível em: <http://rodrigoamaral.net/2012/11/11/qual-a-melhor-ide-para-python/> Acesso em: 17 de out. de 2017.

ARAÚJO, D. C.; RODRIGUES, A. N.; SILVA, C. V. de A.; SOARES, L. S. **O Ensino da Computação na Educação Básica Apoiado por Problemas**: Práticas de Licenciados em Computação**.** In: Anais do XXIII WEI (Workshop sobre Educação em Computação) Garanhuns, 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2015/014.pdf> Acesso em: 25 de nov. de 2017.

BATISTA, E. J. S.; CASTRO, A. A.; BOGARIM, C. A.; LARREA, A. A. **Utilizando o Scratch como ferramenta de apoio para desenvolver o raciocínio lógico das crianças do ensino básico de uma forma multidisciplinar.** Anais do XXI Worshop de Informática na Escola. Ponta Porã, 2015.

BEER, R. **Programação para menores.** Veja. Pag 89. 10 de Julho, 2013.

BURDON, D. **Stronger 'Western' focus in new national curriculum**. Disponível em: < http://www.sunshinecoastdaily.com.au/news/Stronger-Western-focus-in-new-nationalcirriculu/2780704/> Acesso em: 15 de nov. de 2017.

CASTRO, A.; KOSCIANSKI, A. **O uso da Programação Scratch para o Desenvolvimento de Habilidades em Crianças do Ensino Fundamental.** Revista Tecnologias na Educação, Vol.19. Paraná, 2017.

CASTRO, C. T., CASTRO JUNIOR, A., MENESES, C., B., M. e RAUBER, M. **Utilizando Programação Funcional em Disciplinas Introdutórias de Computação**, XI Workshop de Educação em Computação – WEI, Campinas, 2003.

CODE CLUB BRASIL. **Criando um Code Club,** 2017. Disponível em: <http://www.codeclubbrasil.org.br/criando-um-code-club/> Acesso em: 09 de out. de 2017.

COSTAS, R. Em busca de emprego? **Saiba quais os setores que mais resistem à crise**. Disponível em: <http://www.bbc.com/portuguese/noticias/2015/06/150622_emprego_onde_ru>. Acesso em: 15 de nov. de 2017.

DEPARTAMENTO DE ENGENHARIA ELÉTRICA. **Estrutura do curso e carga horária | DEE UFC.**2005. Disponível em: <http://www.dee.ufc.br/graduacao/estrutura-do-curso-e-carga-horaria/>. Acesso em: 11 de dez. de 2017.

DEWAY. **A empresa: tecnologia, experiência e compromisso a serviço de seu negócio,** 2017. Disponível em: <http://www.deway.com.br/a-empresa> Acesso em: 17 de out. de 2017.

DINELLO, R. **Os jogos e as ludotecas**. Santa Maria: Pallotti, 2004

FERRARI, P. A. Lógica de programação e linguagem corporal: **Programação desplugada em sala de aula**, 2017. Disponível em: <https://www.aprendizagemcriativa.info/unplugedd> Acesso em: 20 de out. de 2017.

FERREIRA, C.; GONZAGA, F.; SANTOS, S. **Um Estudo sobre a Aprendizagem de Lógica de Programação por Demonstração**. XVIII WEI (Workshop sobre Educação em Computação) Belo Horizonte, 2010. Disponível em: <http://www.inf.pucminas.br/sbc2010/anais/pdf/wei/st06_03.pdf>. Acesso em: 10 de out. de 2017.

FLORENZANO, C. **Países e cidades do mundo começam a tornar obrigatório o ensino de programação nas escolas**. Disponível em: <http://www.cbsi.net.br/2015/09/paises-e-cidadesdo-mundo-comecam-a-ensinar-computacao-obrigatoriamente.html>. Acesso em: 15 de nov. de 2017.

FUNCEME. **Apresentação**, 2009. Disponível em: <http://www.funceme.br/index.php/menu-institucional> Acesso em: 17 de out. de 2017.

GARCIA, R. E.; CORREIA, R. C. M.; SHIMABUKURO, M. H. **Ensino de Lógica de Programação e Estruturas de Dados para Alunos do Ensino Médio**. In Anais do XXVIII

Congresso da SBC-WEI. Belém, 2008.

GARLET, D.; BIGOLIN, N. M.; SILVEIRA, S. R. **Uma Proposta para o Ensino de Programação de Computadores na Educação Básica.** Santa Maria, 2016.

GERALDES, W. B. **programar é bom para as crianças?** Uma visão crítica sobre o ensino de programação nas escolas. Texto Livre: Linguagem e Tecnologia, vol. 7, num. 2. Goiás, 2014.

GUIMARÃES, A. M.; LAGES, A. C. **Algoritmos e estruturas de dados**. Rio de Janeiro: LTC Editora S.A., 1994.

KOCHAN, S. G. **Programming in C.** 3. ed. Indianapolis: Developer's Library, 2005.

LEAL, A. V. A. **Ensino de Programação no Ensino Médio Integrado:** Uma Abordagem Utilizando Padrões e Jogos com Materiais Concretos. Goiânia, 2014.

LOVE, D. **A Conversation with Linus Torvalds, who built the world's most robust operating system and gave it away for free**, 2014. Disponível em: <http://www.businessinsider.com/linus-torvalds-qa-2014-6>. Acesso em: 31 de out. de 2017.

MIND BENDING. **A história do Python**. Disponível em: <http://mindbending.org/pt/a-historia-do-python > Acesso em: 16 de out. de 2017.

MORPHUS. **Bem vindos a Morphus**, 2017. Disponível em: <http://www.morphus.com.br/#wrapper> Acesso em: 17 de out. de 2017.

MULLER, N. **Diferenças entre copilador e interpretador,** 2009. Disponível em: <http://www.oficinadanet.com.br/artigo/1527/diferencas_entre_copiladores_e_interpretadores > Acesso em: 17 de out. de 2017.

NASCIMENTO, C. S. **Introdução ao Ensino de Lógica de Programação para Crianças do Ensino Fundamental com a ferramenta Scratch.** Rarainópolis, 2015.

NETO, P. **Brincando com as Frações: Sistema de Jogos Educativos**. Canoas: ULBRA, 2001.

OLIVEIRA, B. A. A.; MORAIS, I. S.; ANJOS, E. G.; SOARES, V. G.; FAGUNDES, V. A. C. **Ensino de Linguagem de Programação no Ensino Fundamental e Médio:** Ampliando o Acesso através da EA. Paraíba, 2013. Disponível em: <http://www.prac.ufpb.br/enex/trabalhos/4CIDIPROBEX2013548.pdf> Acesso em: 15 de out. de 2017.

OLIVEIRA, M. L. S.; SOUZA, A. A.; BARBOSA, A. F.; BARREIROS, E. F. S. **Ensino de lógica de programação no ensino fundamental utilizando o Scratch**: um relato de experiência. XXXIV Congresso da Sociedade Brasileira de Computação – CSBC. Pernambuco, 2014.

OLIVEIRA, R. S. **Introdução à Linguagem de Programação Python.** XIV Semana da Computação da UFJF. Juiz de Fora, 2010.

PET – ENGENHARIA ELÉTRICA/UFC . **Eletrônica Digital**, 2014. Disponível em: <http://www.peteletrica.ufc.br/minicursos/>. Acesso em: 11 de dez. de 2017.

PYTHON BRASIL. **Empresas que usam Python.** 2017. Disponível em: <http://www.python.org.br/empresas> Acesso em; 16 de out. de 2017.

PYTHON SOFTWARE FOUNDATION. **Mission Statment Page.** Disponível em: <http://www.python.org/psf> Acesso em: 06 de dez. de 2017.

RAMALHO, L. **Aprenda a Programar.** São Paulo, 1999. Disponível em: <http://edisciplinas.usp.br> Acesso em: 06 de dez. de 2017.

REBOUÇAS, A. D. D. S.; MARQUES, D. L.; COSTA, L. F. S.; SILVA, M. A. A. **Aprendendo a Ensinar Programação Combinando Jogos e Python**. In Anais do XXI Simpósio Brasileiro de Informática na Educação (SBIE). Paraíba, 2010.

SCHÄFER, P. B.; SPERB, B. F.; FAGUNDES, L. C. Squeak Etoys na modalidade 1 para 1:

programação e autoria multimídia no desenvolvimento da conceituação. In Anais XXII SBIE - XVII WIE, Aracaju, 2011.

SCRATCH. **About Scratch,** 2017. Disponível em: <http://scratch.mit.edu/about> Acesso em: 07 de dez. de 2017.

SEBESTA, R. W. **Conceitos de linguagens de programação**. 9. Ed. Porto Alegre: Bookman, 2011.

SETZER, V. W. Computadores na educação: por quê, quando e como. In: SETZER, V. W. **Meios Eletrônicos e Educação**: uma visão alternativa, 2. ed. São Paulo: Escrituras, 2002, p. 85-134. Disponível em: <http://www.ime.usp.br/~vwsetzer/PqQdCo.html>. Acesso em: 10 nov. 2017.

SILVEIRA, S. R. **Estudo e Construção de uma ferramenta de autoria multimídia para a elaboração de jogos educativos.** Dissertação de Mestrado POA-PPGC UFRGS, 1999.

WILLRICH, R. INE5602**: Introdução à Informática**. Florianópolis, 2000. Disponível em: <http://algol.dcc.ufla.br/~monserrat/icc/Introducao_linguagens.pdf> Acesso em: 12/10/2017.

APÊNDICE A – FICHA DE INSCRIÇÃO PARA O CURSO INTRODUTÓRIO À LINGUAGEM DE PROGRAMAÇÃO *PYHTON GAMES*.

Inscrições - Curso Introdutório à Linguagem de Programação

Todas as respostas são de caráter obrigatório, elas servirão como critério de seleção para os participantes do curso.

*Obrigatório

1. Nome: *

2. E-mail: *

3. Telefone: *

4. Série: *

- ◯ Ensino Fundamental
- ◯ 1º Ano - Ensino Médio
- ◯ 2º Ano - Ensino Médio
- ◯ 3º Ano - Ensino Médio
- ◯ Outro:

5. Escola:

6. Tipo de Escola: *

- ( ) Pública
- ( ) Privada
- ( ) Outro: _____

7. Sua escola possui laboratório de informática? *

- ( ) Sim
- ( ) Não

8. Caso sim, com qual frequência os alunos visitam o laboratório de informática?

- ( ) Menos de 1 vez ao mês
- ( ) 1 vez ao mês
- ( ) 2 Vezes ao mês
- ( ) Mais de 2 vezes ao mês

9. O quanto você se interessa por Matemática? (1 - baixo e 5 - alto) *

1	2	3	4	5
( )	( )	( )	( )	( )

10. O quanto você se interessa por Computadores? (1 - baixo e 5 - alto) *

1	2	3	4	5
( )	( )	( )	( )	( )

11. O quanto você se interessa por Jogos? (1 - baixo e 5 - alto) *

1	2	3	4	5
( )	( )	( )	( )	( )

12. Já realizou algum curso na Seara da Ciência? *

- [ ] Sim
- [ ] Não

13. Já tem conhecimento de alguma linguagem de programação? *

- [ ] Sim
- [ ] Não

14. Com qual frequência usa computador? *

- [ ] Não uso
- [ ] Menos de 2 horas por dia
- [ ] Entre 2 e 5 horas por dia
- [ ] Mais de 5 horas por dia

15. Quando está usando o computador, que atividade realiza na maior parte do tempo? *

- [ ] Jogos
- [ ] Assistir Séries/Filmes/YouTube
- [ ] Estudar
- [ ] Redes Sociais (Facebook, twitter, G+)
- [ ] Ler/Assistir reportagens e/ou documentários
- [ ] Não uso computador
- [ ] Outro: _____

10. Escreva algumas linhas sobre o que você espera aprender no curso ofertado: *

SELEÇÃO

Os alunos selecionados serão noticiados via e-mail no dia 11/09/2016.
Para maiores informações: matheushp7@gmail.com.

APÊNDICE B – TESTE 1: AVALIAÇÃO DO CURSO

SEARA DA CIÊNCIA
AVALIAÇÃO FINAL – PYHON

Monitor: Matheus Roque

1. Gostou da metodologia do curso (uso de apostila, data show e computadores, *dropbox* e *whatsapp*)

   ( )Pouco ( ) Médio ( )Muito
2. Gostou da linguagem de programação utilizada (Python)?
   ( )Pouco ( ) Médio ( )Muito

3. O quanto gostou da didática do monitor?
   ( )Pouco ( ) Médio ( )Muito

4. Gostou da sala de aula (Laboratório de Informática da Seara da Ciência)?
   ( )Pouco ( ) Médio ( )Muito

5. Gostou dos horários e dias da semana (Ter/Qua/Qui– 14:00 às 17:00)?
   ( )Pouco ( ) Médio ( )Muito

6. O quanto o curso mudou sua forma de ver o mundo?
   ( )Pouco ( ) Médio ( )Muito

7. Quão grande é seu desejo de seguir na área de programação (faculdade, curso técnico, etc.)?
   ( )Pouco ( ) Médio ( )Muito

8. Quais as chances de você recomendar o curso para outros amigos?
   ( )Pouco ( ) Médio ( )Muito

9. O quanto você está gostando do curso?
   ( )Pouco ( ) Médio ( )Muito

APÊNDICE C – TESTE 2: AVALIAÇÃO DO ALUNO

SEARA DA CIÊNCIA
AVALIAÇÃO FINAL – PYHON
Monitor: Matheus Roque

Aluno: _____ Data: __/__/__

1. Descreva, com suas palavras, as seguintes expressões:

   a) Linguagem de Programação:

_____
_____
_____
_____

   b) Algoritmo:

_____
_____
_____
_____

2. Usando o computador, desenvolva um programa que calcule o IMC (Índice de Massa Corpórea) de uma pessoa. O programa deve pedir: Massa e Altura e realizar o seguinte cálculo:

$$IMC = \frac{Massa}{Altura^2}$$

No final, o programa deve informar ao usuário o IMC e a classificação do mesmo, segundo a tabela abaixo:

IMC	Classificação		
< 16	Magreza grave	30 a < 35	Obesidade Grau I
16 a < 17	Magreza moderada	35 a < 40	Obesidade Grau II (severa)
17 a < 18,5	Magreza leve	> 40	Obesidade Grau III (mórbida
18,5 a < 25	Saudável		
25 a < 30	Sobrepeso		

3. Desenvolva um programa que desenhe um retângulo na tela, onde o comprimento e a largura são dados pelo usuário.

APÊNDICE D – QUESTIONÁRIO: “PROGRAMAÇÃO É PARA TODOS?”

Programação é para todos?

Formulário criado para obter a opnião de estudantes dos cursos de tecnologia da UFC (ou outras) sobre as dificuldades de se aprender programação.

1. **Estuda em qual Universidade?**
   Marcar apenas uma oval.

   ◯ UFC

   ◯ Outro: _____

2. **Qual curso de graduação?**
   Marcar apenas uma oval.

   ◯ Eng. Petróleo

   ◯ Eng. Ambiental

   ◯ Eng. Energias Renováveis

   ◯ Eng. Meio Ambiente

   ◯ Eng. Civil

   ◯ Eng. Computação

   ◯ Eng. Alimentos

   ◯ Eng. De Minas

   ◯ Eng. De Pesca

◯ Eng. De Produção

◯ Eng. De produção Mecânica

◯ Eng. De Software

◯ Eng. Telecomunicações

◯ Eng. Teleinformática

◯ Eng. Elétrica

◯ Eng. Mecânica

◯ Eng. Metalúrgica

◯ Eng. Química

◯ Outro: _____

3. Realizou alguma disciplina que necessitasse do uso de linguagem de programação?

◯ SIM *ir para perguntar 6.*

◯ NÃO *ir para pergunta 4.*

4. Teve contato com programação fora da Universidade?

◯ SIM *ir para perguntar 5.*

◯ NÃO *Pare de preencher este formulário*

5. Onde realizou esse contato?
Marque todas que se aplicam.

☐ Escola (Ensino Fundamental)

☐ Escola (Ensino Médio)

☐ Projetos fora da escola

☐ Curso Técnico

☐ Em casa

☐ Outros

6. **Em sua opinião: Qual o nível médio de dificuldade que essas disciplinas possuem? (1 – Muito Fácil; 5 – Muito Difícil)**
   Marcar apenas uma oval.

7. **Você teve algum contato com programação antes de ingressa na universidade?**
   Marcar apenas uma oval.

   ◯ SIM *ir para perguntar 8.*

   ◯ NÃO *ir para a pergunta 9.*

8. **Onde obteve esse contato?**
   Marque todas que se aplicam.

   ☐ Escola (Ensino Fundamental)

   ☐ Escola (Ensino Médio)

   ☐ Projetos fora da escola

   ☐ Em casa

   ☐ Outros

9. **O ensino de programação durante o ensino básico (Fundamental e Médio) ajudaria na compreensão de disciplinas que usam essa ferramenta dirante a Universidade?**
   Marque apenas uma oval.

   ◯ SIM

   ◯ NÃO

Obrigado pelo tempo disponibilizado para responder esse questionário.

APÊNDICE E – APOSTILA DESENVOLVIDA NA APLICAÇÃO DO ESTUDO DE CASO

UNIVERSIDADE FEDERAL DO CEARÁ
SEARA DA CIÊNCIA

CURSO INTRODUTÓRIO À LINGUAGEM DE PROGRAMAÇÃO

ALUNO:_____

MATHEUS MAIA ROQUE

SETEMBRO DE 2016
FORTALEZA – CE

Sumário

1. INTRODUÇÃO

O computador é hoje uma das ferramentas de trabalho mais utilizada no mundo inteiro. É praticamente impossível se imaginar no mercado de trabalho sem conhecimentos básicos sobre essa máquina. Usamos diversos programas para realizar nossas tarefas, desde o *Pacote Office* (*Word, Excel, PowerPoint*) até reprodutores de música e vídeo. No entanto você já se perguntou como esses programas são feitos? Como que o computador realiza essas tarefas? Quem cria esses programas?

Todas essas perguntas se direcionam para uma única resposta: Programação. É através das linguagens de programação que o ser humano consegue criar programas, jogos e aplicativos para os computadores e assim serem utilizados por pessoas em suas diversas áreas de trabalho.

Sendo assim, a programação se tornou uma ferramenta poderosa para soluções de problemas nas mais diversas áreas. Seja na área de tecnologia, seja na área de saúde, ter conhecimentos de programação nada mais é que ter o poder de minimizar trabalho, aumentar eficiência e criar soluções inovadoras.

Pensando nisso, foi-se desenvolvido esse curso de programação básica usando linguagem *Python* e voltando-se para a aplicação de jogos de estilo *Arcade*. Aqui o aluno de ensino médio (preferencialmente) terá um primeiro contato com essa ferramenta, além de despertar e aprimorar habilidades como raciocínio lógico e matemática.

A apostila é baseada no trabalho *Program Arcade Games With Python And Pygam*e criado pelo Dr. Paul Vincent Craven e disponível em http://programarcadegames.com/index.php.

1.1. LINGUAGENS DE PROGRAMAÇÃO

Linguagem: “qualquer meio sistemático de comunicar ideias ou sentimentos através de signos convencionais, sonoros, gráficos, gestuais etc”.

A linguagem de programação é então uma forma de comunição entre o ser humano e o computador onde o primeiro, exprime sobre alguma forma sua ideia e o segundo consegue captar e executar o que foi pedido. Assim como todas as outras linguagens, a programação é constituída de regras gramaticais que servem para formalizar e organizar as informações.

Existem várias linguagens de programação, no entanto o computador entende apenas uma delas: A linguagem de Máquina. Essa linguagem é formada por bytes de informação que é considerada uma linguagem de baixo nível para o programador devido a sua complexidade.

```
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000100000000001000000000
00000000000011100001111110111010000011100000000010110100000000100
11100110100100001101110000000000101001100110011010010000010101010
00011010000110100101110011001000000111000001110010011011110110 0
11101110010011000010110110100100000011000110110000101101110011 0
11100110111101110100001000000110001001100101001000000111001001 1
```

Figura 17 -Exemplo de Linguagem de Máquina.

Com o tempo, novas linguagens foram criadas com o intuito de facilitar o trabalho do programador. No entanto essas linguagens precisam de tradutores para transforma-las em linguagem de máquina usando um copilador ou interpretador. É o caso, por exemplo, da Python que vamos trabalhar nesse manual. Ela é considerada uma linguagem de alto nível devido a sua maior proximidade com o programador.

1.2. BAIXANDO ARQUIVOS ESSENCIAIS

O curso aqui desenvolvido será ministrado completamente em linguagem Python, por isso, será necessário fazer o download de alguns itens para desenvolver nossos programas.

1º - Baixo o executável do python através do link:
ProgramArcadeGames.com/python-3.3.3.msi
2º - Baixe a biblioteca de jogos Pygame:
ProgramArcadeGames.com/pygame-1.9.2a0.win32-py3.3.msi
3º - Baixe o editor Wing (IDLE):
wingware.com/downloads/wingide-101/

1.3. ALGUMAS INFORMAÇÕES IMPORTANTES

Antes de iniciar nossos programas, devemos ter em mente algumas instruções básicas sobre o python, IDLE e biblioteca pygame.

A linguagem python que você baixou (arquivo 1 acima mencionado) já é por si só suficiente para realizar nossos programas, no entanto, trabalhar com o executável do programa original é dificultoso. Por isso, fazemos o uso do IDLE Wing (arquivo 3). Ele nos ajudará a organizar melhor nossas linhas de código e a detectar possíveis erros.

O arquivo 2 que baixamos é uma biblioteca usada para facilitar nossa criação de jogos. Dentro desse arquivo podemos adquirir vários comandos já condensados para criarmos desenhos, animações e controle.

O monitor deve em sala mostrar o passo a passo de como fazer o download dos arquivos, bem como a instalação de cada um deles. No caso de dúvidas não hesite em perguntar. Você deverá realizar os mesmos procedimentos em casa.

2. ALGORITMO

O uso de Algoritmos é recorrente em todos os momentos do nosso dia a dia, desde comprar um lanche na padaria, até responder uma prova de matemática na escola. No entanto, muitos associam o termo Algoritmo apenas à computação e não o percebem a sua volta.

Podemos definir então Algoritmo como uma receita para realizar determinada tarefa, receita esta clara e objetiva com todos os procedimentos bem estabelecidos ou *‘ conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas. ’*

2.2. EXEMPLO: ALGORITMO PARA FAZER MACARRÃO

Start

1 Abra a válvula de gás do fogão;
2 Acenda o fogo usando um fósforo;
3 Coloque uma panela com 1 l de água no fogão acesso;
4 Espere 5 minutos;
5 Verifique se a água está fervendo
Se não:
Espere mais 2 min e volte para o passo5;
Se sim:
Adicione o macarrão;
Adicione 1 colher de sopa de sal;
Adicione temperos a gosto
Espere mais 7 min;

6 Verifique se o macarrão está mole usando um garfo;
Se sim:
Retire o macarrão e vá para o passo 7;
Se não:
volte para o passo 6
7 Escorra o macarrão;
8 Sirva o macarrão

End

Escolha uma atividade típica do seu dia-a-dia e construa um Algoritmo que mostre a forma correta de realizar a tarefa. Lembre-se, é importante que os procedimentos sejam claros e que ofereçam suporte para as dúvidas que surgirem durante a execução do procedimento.

2.3. O USO DE FLUXOGRAMAS

Da mesma forma que podemos usar texto para escrever nossos procedimentos para realização de uma tarefa, podemos ainda utilizar de representações gráficas, como figuras geométricas, para facilitar o processo de construir um algoritmo. Veja a seguir o procedimento de fazer um macarrão através de um **Fluxograma**.

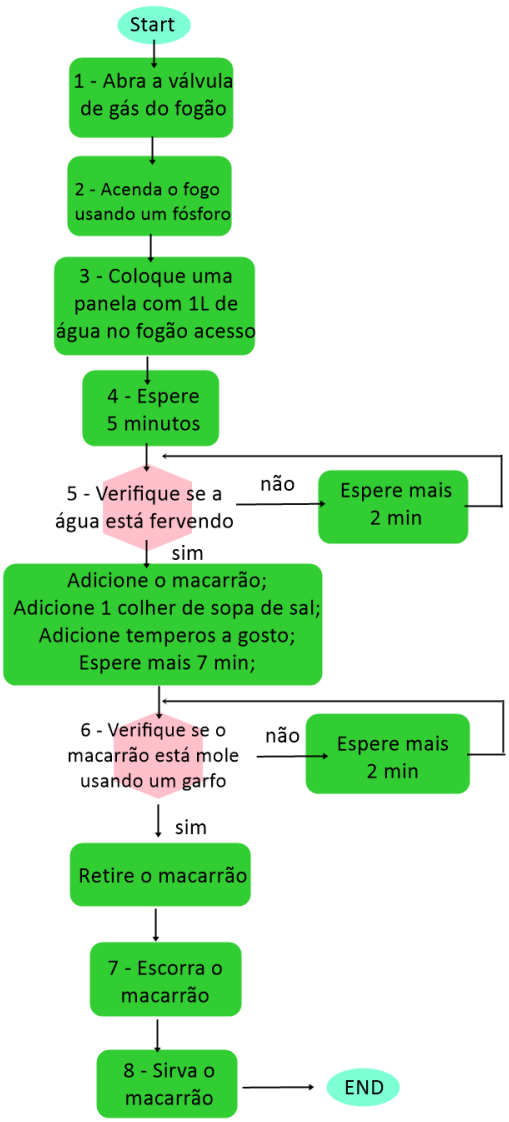


Figura 18 - Fluxograma de um preparo de macarrão.

Usando como exemplo a Figura 18 - Fluxograma de um preparo de macarrão., mostre um fluxograma baseado nos procedimentos que você criou no item 2.2. Fique a vontade para escolher qualquer forma geométrica para indicar os procedimentos bem como as tomadas de decisão.

3. HELLO WORLD

Um mito entre estudantes de programação diz que: *"O primeiro programa de um programador deve ser escrever HELLO WORD, se isso não for feito, o programador terá anos de azar."* Para não corrermos esse risco aprenderemos de início o comando para escrever, e cumpriremos as normas para que não soframos com anos de 'azaração'.

3.2. O COMANDO PRINT

O comando para escrever algo na tela do seu programa é:

```
print("Hello World.")
```

Para escrever algo na sua tela é necessário apenas escrever o comando Print seguido de parênteses e aspas duplas, a frase a ser escrita deve estar entre as aspas e os parênteses.

Feito seu primeiro programa, tente escrever outras frases. Teste escrever frases sem o uso das aspas duplas e dos parênteses. O que ocorre?

Escreva em sua tela a expressão *5+4* usando as aspas. Após ver o resultado, faça o mesmo procedimento sem usar as aspas. Qual a diferença?

3.3. ESCREVENDO RESULTADOS DE EXPRESSÕES

Você pode perceber que há uma diferença em usar o comando Print com aspas e sem aspas quando se trata de expressões matemáticas.

Quando você utiliza-se de aspas, o comando irá entender que estamos tratando Strings, ou seja, o conteúdo dentro dos parênteses é do tipo texto. Quando retiramos as aspas, o comando entende como uma expressão matemática, e escreverá na sua tela o resultado da expressão.

Por exemplo:

```
print("2+6") → Tela: 2+6
print(2+6)   → Tela: 8
```

Caso você tente escrever *Hello World* sem o uso de aspas, o comando indicará um erro, visto que o conteúdo dentro dos parênteses não faz referência a uma expressão matemática.

3.4. ESCREVENDO ITENS DIFERENCIADOS

Algumas vezes necessitamos escrever frases, números e resultados de expressões em um mesmo comando *print* para economizar linhas e tornar o programa mais organizado. Para isso, podemos brincar com as aspas e separar cada item desejado por vírgulas.

Por exemplo:

Imagina que temos 5 pessoas na nossa família com as seguintes idades:

Pai – 66 anos
Mãe – 54 anos
Eu – 23 anos
Irmão – 25 anos

Gato – 2 anos

E quero escrever na minha tela a seguinte frase:

A soma das idades dos membros da minha família é: ____

Só que em vez dos traços quero escrever a soma das idades. Para que eu não precise fazer a soma mentalmente ou através de uma calculadora, posso fazer o seguinte programa que realizará essa tarefa:

```
print("A soma das idades dos membros da minha família é:", 66 + 54 + 23 + 25 + 2)
```

Caso eu queira escrever apenas a minha idade, como faria?

3.5. ESCREVENDO COMENTÁRIOS

Tudo que escrevemos no nosso IDLE do *Python* é interpretado de alguma forma, processado e liberada uma informação para nossa terra. Caso escrevamos algo que o IDLE não consegue interpretar, esse informará uma mensagem de erro. Assim, temos que estar sempre atentos nos comandos e na forma correta de usá-los.

No entanto, existe uma forma de 'enganar' o nosso IDLE. Esse processo consiste em usar o símbolo # para escrever algo que não queiramos que seja lido.

Exemplo:

```
# Esse é um comentário
# O computador vai ignorar o que vem depois da # #(Hastag).
print("Já isso não é um comentário")
      print("O computador irá tentar interpretar isso.”)
```

Escreva algo como comentário em seu programa e comprove.

3.6. VARIÁVEIS: ONDE VALORES SÃO GUARDADOS

Em todos os jogos que existem, vários valores precisam ser guardados e depois usados. Valores como: Nível de XP e HP de uma personagem, quantidade de itens acumulados, distância percorrida, etc.

A forma que usamos para guardar esses valores são as **variáveis**. Elas já são bem comuns em várias aplicações da matemática. Aqui, usaremos as variáveis com o mesmo propósito: Armazenar (guardar) valores e resultados de alguma operação para serem usados mais adiante.

Exemplo:

Vamos analisar o seguinte programa:

```
x = 5
print(x)
```

Na primeira linha, estamos armazenando o valor 5 na variável x através do ***Operador de Atribuição =*** . Ou seja, existira certa caixa chamada x onde guardamos o número 5. Se em qualquer momento do programa, usarmos essa caixa, ela mostrará o que está contido dentro, neste caso seria o número 5. Redija o programa em seu computador e veja o resultado. Tente outros valores. Use aspas em torno do x. O que mudou?

Podemos realizar operações com o valor x. E, como seu próprio nome diz, ele poderá ter seu conteúdo variado.

Exemplo:

```
x = 5
x = x + 1
print(x)
```

Agora, o resultado na sua tela não será mais 5 e sim o valor 6.

Podemos escolher diversos nomes para as variáveis, no entanto precisamos seguir algumas regras na hora de designar essas nomenclaturas.

1 – Não se deve usar nomes reservados do programa, como por exemplo *print*.

2 – Caso o nome seja composto, este não deve ser separado pó espaço, normalmente usa-se _ (hífen) ou se junta as duas palavras (menos comum).

3 – É obrigatório iniciar a palavra com letra, não podendo iniciar com números ou caracteres especiais.

4 – Usam-se normalmente letras minúsculas na declaração da variável.

Obs1: normalmente escolhe-se um nome para a variável de acordo com a função que ela vai exercer no programa. Por exemplo, se uma variável vai guardar a soma das idades da família, pode-se usar o nome '*soma_idades'*.

Obs2: O computador diferencia letras maiúsculas de minúsculas na declaração de uma variável. Assim, uma variável com nome *Casa* é diferente de uma com nome *casa.*

No quadro a seguir, avalie se as variáveis foram declaradas de forma correta perfeita, correta imperfeita ou incorreta.

Nomenclatura	Correta Perfeita	Correta Imperfeita	Incorreta
Soma_Idades			
SomaIdades			
Soma_idades			
X			
y			
8_Valores			
8valores			
%altura_irmao			
Altura_irmão			
Altura_irmão			

3.7. OPERADORES

Todas as operações básicas da matemática podem ser realizadas em um programa, bem como outros tipos de operações que podem ser úteis durante a criação de seus jogos. Para isso, tem-se abaixo uma imagem com os símbolos utilizados para realizar essas operações. Tente você mesmo utilizar algumas delas usando o comando *print* e veja os resultados.

operator	operation	example equation	example code
+	addition	$3 + 2$	a = 3 + 2
-	subtraction	$3 - 2$	a = 3 - 2
*	multiplication	$3 \cdot 2$	a = 3 * 2
/	division	$\frac{10}{2}$	a = 10 / 2
//	floor division	N/A	a = 10 // 3
**	power	2^3	a = 2 ** 3
%	modulus	N/A	a = 8 % 3

Figura 19 - Lista de operadores

Quando se desejar realizar uma equação matemática usando a linguagem *python* é necessário estabelecer parênteses para delimitar que operação será realizar primeiro, e qual será realizada depois.

Exemplo:

```
media = 90 + 86 + 71 + 100 + 98 / 5
media = (90 + 86 + 71 + 100 + 98) / 5
```

As duas expressões acima oferecem diferentes valores. Quais são eles? O que houve de diferente?

Além dos operadores listados acima, podemos ainda realizar operações mais complexas como o Seno e o Cosseno (operações trigonométricas). No entanto, para tal, deve-se importar a *biblioteca* **math** no seu programa. Para isso, basta adicionar, no inicio do programa, o comando:

```
from math import *
```

Obs: As operações trigonométricas realizadas nessa biblioteca usam valores em radianos. Logo, se quiser saber o seno de 30º, deve-se fazer:

Sin((2*pi*30)/360) .

3.8. RECEBENDO INFORMAÇÕES

Provavelmente, será necessário em nossos programas receber informações do usuário e trabalhar essas informações dentro do programa. Por exemplo, quando criamos um jogo e estamos frequentemente recebendo informações do teclado e do mouse que o jogador está enviando.

A função que recebe essas informações é a função **input**. No entanto, essa função é feita para receber apenas strings. Por isso, após a função *input*, normamente converte-se a informação para outros tipos de objetos como: inteiros, float, etc.

Exemplo:

```
#Programa que recebe duas idades e soma elas.
idade1 = input ("Digite sua idade: ")
idade1 = float (idade1)
idade2 = input ("Digite a idade de seu irmão: ")
idade2 = float (idade2)
total = idade1+idade2
print (total)
```

Tente realizar o mesmo procedimento sem usar a conversão de *string* para *float*. O que ocorreu?

3.9. PROJETANDO

Agora que você teve conhecimento básico sobre as operações, vamos usa-lo para projetar seu primeiro programa. Escola um dos projetos sugeridos abaixo, e monte seu programa da forma mais organizada possível.

CALCULADORA DE EQUAÇÕES

1 – Calculadora que mostre a velocidade de um objeto:

Equação:

$$V = \Delta S/\Delta t$$

O programa deve pedir ao usuário o deslocamento e o tempo, e dar como resposta a velocidade.

2 – Calculadora da renda per capta de uma família:

Equação:

$$\text{Renda} = \frac{\text{Salário(Pai+Mãe+Filho1+Filho2)}}{\text{Número de pessoas}}$$

O programa deve pedir o salário do pai, da mãe e dos irmãos e mostrar a renda da família.

Obs: Nesse momento o programador estabelecerá o número de pessoas da família. Mais adiante, veremos como fazer um programa que perguntará a quantidade de pessoas na família e o salário de cada uma delas.

3 – Calculadora para a média de notas de um aluno durante o ano.

Essa calculadora leva o mesmo princípio da calculadora 2, no entanto, o programa pede as 4 notas do ano e divide a soma destas por 4.

4. CONDIÇÕES

"Se a água estiver fervendo, adicione o macarrão."
"Se a água não estiver fervendo, espere mais 2 minutos."

Nas duas frases acima, respondemos uma pergunta com SIM ou NÃO, VERDADEIRO OU FALSO. A pergunta era: "A água esta fervendo?". Caso a resposta seja SIM/VERDADEIRO adicionaremos o macarrão, caso a resposta seja NÃO/FALSO, esperaremos mais 2 minutos. Em programação chamamos essa escolha de **Estruturas Condicionais** ou do Inglês *Conditional Statement.* Veremos a seguir uma estrutura que realiza esse procedimento, o **IF**.

COMANDO IF

Para usar o comando IF deve-se seguir a seguinte regra:

IF (*condição***)**:
    Tarefas a serem realizadas

É de grande importância, que as tarefas a serem realizadas dentro do IF sejam espaçadas à esquerda.

Vamos fazer uma analogia ao macarrão:

IF(*água fervendo*):
    Adicione o macarrão
IF(*água não fervendo*):
    Espere mais 2 minutos

Caso a condição dentro do IF seja verdadeira, as atividades espaçadas que seguem abaixo serão realizadas.

Importante saber que se for esperada mais de uma atividade a ser realizada pelo mesmo IF, estas devem estar igualmente espaçadas.

Exemplo:

```
a = 4
b = 5
# Comparações Básicas
if a < b:
      print("a é menor que b")
if a > b:
      print("a é maior que b")
print("Done")
```

A tabela a seguir mostra os principais sinais de comparações:

Tabela 4 - Sinais condicionais

Sinal	Descrição
<	Menor que
>	Maior que
<=	Menor ou igual que
>=	Maior ou igual que
==	Igual a
!=	Diferente de

É muito importante saber a diferença entre os símbolos '=' e '=='.

= Este símbolo serve para atribuir um valor a determinada variável. Exemplo: *idade_aluno = 24*

== Este símbolo serve para comparar dois valores. Exemplo: a==b (a é igual a b?)

4.2. CONDIÇÕES MÚLTIPLAS

Muitas vezes, queremos realizar uma mesma tarefa para condições diferentes. Por exemplo:

Caso 1:
Vamos jogar um dado.
Se o resultado for 1; 3 ou 5 – você ganha
Caso contrário, eu ganho o jogo.
Caso 2:
Vamos escolher quatro alunos: Aluno A, Aluno B, Aluno C e Aluno D.
Se A for mais velho que B e C for mais velho que D, você ganha o jogo
Caso contrário, eu ganho o jogo.

Observe que, no Caso 1, uma tarefa será realizada casa **ocorra um dos resultados** (valor 1, 3 ou 5). Já no Caso 2, a tarefa será realizada apenas se **ambas as condições foram obedecidas**.

Em programação, usamos o comando **OR** (em português significa *ou*) para o primeiro caso, e usamos **AND** (em português *e*) para o segundo caso.

Veja no exemplo abaixo como esses comandos podem ser usados dentro do comando IF.

```
if a < b and a < c:
      print ("a é menor que b E c ao mesmo tempo")
if a < b or a < c:
      print("a é menor que b OU menor que C (ou ambos)")
```

4.3. COMANDO *ELSE* E *ELIF*

Abaixo se encontra um código que pergunta a temperatura do ambiente. Caso seja maior que 35ºC o programa emitirá uma mensagem "*Nossa, que quente!*":

```
temperatura = int(input("Qual a temperature ambiente? "))
if temperatura > 35:
      print ("Nossa, que quente")
print("Fim")
```

Veja que, se a temperatura for maior menor ou igual a 35, o programa emitirá apenas a frase "*Fim*". Podemos modificar o programa para que ele emita a frase "*Não está tão quente*" caso a temperatura seja menor que 35 usando o comando **ELSE.** Veja o exemplo:

```
temperatura = int(input("Qual a temperature ambiente? "))
if temperatura > 35:
      print ("Nossa, que quente")
else:
      print("Não está tão quente")
print ("Done")
```

Podemos ainda adicionar uma condição para determinar uma faixa de temperatura onde emitira a frase *"Bruuu, que frio!"*, usando o comando **ELIF**.

```
temperatura = int(input("Qual a temperature ambiente? "))
if temperatura > 35:
      print ("Nossa, que quente")
elif temperatura < 20:
      print("Bruuu, que frio!")
else:
      print ("Não está tão quente")
print ("Done")
```

Muito cuidado na hora de usar os comandos IF, ELSE e ELIF. O comando IF sempre deve vir primeiro com a condição principal. Em seguida, podem-se adicionar vários comandos ELIF com condições secundárias. Por último, deve-se por o comando ELSE que retrata as condições que não foram declaradas.

Vamos criar um programa que faça o seguinte:
Peça a média do aluno.
Caso a média seja maior que 7 deve-se emitir na tela *"Parabéns, você foi aprovado."*. Caso a média esteja entre 4 e 7 emitir *"Você está de recuperação!"*. Caso seja menor que 4 "*Que pena,você reprovou"*.

4.4. COMPARANDO STRINGS

Sabe quando *logamos* no *facebook* e ele pede uma senha para confirmar seu *Log in*? Muito simples, podemos fazer isso com comparações de *strings.* Veja o exemplo abaixo:

```
user_name = input("Qual o seu nome? ")
if user_name == “Roque”:
      print("Que nome legal!!!
else:
      print("Seu nome é sem graça!")
```

Observe que:

-

no comando *input* não foi necessário fazer nenhuma conversão.

no comando *IF* usaram-se as aspas na palavra *Roque* para indicar que é uma *string.*

Perceba também que o usuário poderia escrever o nome *roque* e não *Roque.* Caso o R esteja minúsculo, o programa entenderá como uma palavra diferente. Como resolver esse problema? Usando o comando **.lower()**.

```
user_name = input("Qual o seu nome? ")
if user_name.lower() == “roque”:
      print("Que nome legal!!!
else:
      print("Seu nome é sem graça!")
```

4.5. PROJETANDO

TESTE ONLINE

Para esse capítulo 4 o projeto é criar um teste online.

Escolha 5 perguntas que você gostaria de por no seu teste. O ideal é que essas perguntas tenham diferentes tipos de respostas (números, palavras, letras, etc.). Seu programa deve emitir a pergunta para o usuário, receber a resposta, e informar se o usuário respondeu *Correto* ou *Incorreto*. A seguir, um exemplo:

```
1  #Teste Online
2
3  questao1 = float(input("Quantos Livros tem a série Jogos Vorazes? "))
4  if questao1 == 3:
5      print("Correto")
6  else:
7      print("Incorreto")
8
9  questao2 = input("A música Toxic pertence a que cantora POP? ")
10 if questao2.lower() == "britney":
11     print("Correto")
12 else:
13     print("Incorreto")
```

5. REPETIÇÕES

No capítulo anterior você aprendeu a usar o comando IF que é uma estrutura condicional dentro da programação. Nesse capítulo, será estudada outra estrutura fundamental para que um programador consiga realizar seus trabalhos com eficiência, são as chamadas **Estruturas de Repetição** ou do inglês *Loop.*

O objetivo do Loop é realizar uma determinada tarefa uma numa quantidade de vezes determinada pelo programador. Por exemplo:

Caso 1 :

Você irá escrever na lousa 5 vezes "Não devo contar mentiras".

Caso 2

Enquanto não der 17:00 h você vai escrever numa folha de papel "Não devo contar mentiras".

5.2. COMANDO FOR

Entender o comando **FOR** de início pode ser uma tarefa difícil, mas bastam alguns exemplos e práticas para que o conceito fique bem gravado na cabeça do programador que está iniciando. A estrutura do comando é dada por:

For *variável* **in** *loop:*

Tarefa

Você pode dar qualquer nome para a variável que vai ser usada no comando *FOR,* no entanto é mais comum usarmos a letra "*i*" para essa função, pois ela remete a palavra incremento.

Para o *loop* usa-se normamente o comando **range(*n*),** onde '*n*' é o número de vezes que se queira realizar a tarefa. Veja o exemplo a seguir:

```
for i in range(5):
    print("Programar é muito fácil!")
```

```
Saída:
Programar é muito fácil!
Programar é muito fácil!
Programar é muito fácil!
Programar é muito fácil!
Programar é muito fácil!
```

Observe que as tarefas a serem realizadas pelo comando *FOR* devem estas espaçadas a esquerda. Pode-se inclusive realizar mais de uma tarefa. Podemos ainda adicionar um comando *FOR* dentro de outro comando *FOR*.


```
for i in range(3):
      print ("a")
      for j in range(3):
      print ("b")
print("Done")
```

Qual a saída do programa?

A variável *'i'* assume os valores do Loop para realizar as atividades. Quando usamos, por exemplo, o *range(3),* a variável *'i'* esta assumindo os valore: 0, 1 e 2. O comando range indica quantos números a variável vai assumir, mas começando do número '0'. Vamos fazer um teste:

Faça um programa com o seguinte script:

```
for i in range(5):
      print(i)
```

Qual saída você obteve? Experimente usar outros números dentro do comando *range().*

Qual a diferença do programa acima para o seguinte:

```
for i in range(5):
      print("i")
```

Se você não quiser começar o valor da variável de '0', você pode fazer a seguinte modificação no programa:

```
for i in range(1,6):
      print(i)
```

Veja que dessa vez a saída será os números de 1 até 6 (sem incluir o 6).

Caso queira imprimir números com um passo (*step*) fixo, podemos acrescentar ainda um terceiro número dentro do comando *range()* para isso.

```
for i in range(1,10,2):
      print(i)
```

Saída:

1
3
5
7
9

Em vez de usar o comando *range()* podemos determinar diretamente os números que queremos fazer o *Loop*, usando colchetes [].

```
for i in [2,6,4,2,4]:
    print(i)
```

Saída:

2
6
4
2
4

5.3. PRATICANDO COM O COMANDO FOR

A seguir temos uma lista com algumas questões utilizando o comando *FOR* que você tentou aprender no item 5.1. Tente prever o que cada programa emitirá na tela. Em seguida, faça os programas utilizando o IDLE e veja se você acertou a previsão. Sei que o comando *FOR* pode bagunçar um pouco sua cabeça, mas não desista da programação! Um pouco mais de treino e tudo ocorrerá bem.

Programa 1:

```
for i in range(10):
    print("Hi")
```

Programa 2:

```
for i in range(5):
    print("Hello")
    print("There")
```

Programa 3:

```
for i in range(5):
    print("Hello")
print("There")
```

Programa 4:

```
for i in range(10):
    print(i)
```

Programa 5:

```
for i in range(1, 11):
    print(i)
for i in range(10):]
    print(i + 1)
```

Programa 6:

```
for i in range(2, 12, 2):
```

```
        print(i)
for i in range(5):
        print((i + 1) * 2)
```

Programa 7:

```
for i in range(10, 0, -1):
        print(i)
```

Programa 8:

```
for i in [2, 6, 4, 2, 4, 6, 7, 4]:
        print(i)
```

Programa 9:

```
for i in range(3):
        print("a")
        for j in range(3):
                print("b")
```

Programa 10:

```
a = 0
for i in range(10):
        a = a + 1
print(a)
```

Programa 11:

```
sum = 0
for i in range(1, 101):
        sum = sum + i
```

5.4. COMANDO WHILE

Vimos no início do capítulo 5 dois casos onde se usa uma repetição de tarefa. No primeiro caso, fica determinado quantas vezes a pessoa deve escrever na lousa a frase "*Não devo contar mentiras*", já no segundo caso, a quantidade de vezes não é determinada, mas a atividade sera repetida até um determinado ponto.

Esses dois casos mostram muito bem a diferença de usar os camando *FOR* e o comando *WHILE*. Pelo que vimos na unidade 5.1. o commando FOR sera usado no primeiro caso, ou seja sempre que soubermos a quantidade de vezes que sera realizada a atividade.

Já o comando ***WHILE*** será usado sempre quando não sabemos a quantidade de vezes que a tarefa sera executada, mas sabemos até quando ela será. É o que ocorre no caso 2, por exemplo.

Veja a estrutura do comando a seguir:

while *variável condição***:**

tarefa

Diferente do comando *FOR*, é necessário dar um valor para a variável antes de usar o comando *while*. O comando não irá atribuir um valor para ela, e acusará um erro *"Variável não existe"* se ela não for declarada. Veja a seguir alguns exemplos.

Exemplo 1:

```
i = 0
while i < 4:
  print(i)
  i = i + 1
```

Saída:

```
0
1
.... 2
.... 3
```

No exemplo 1 declaramos a variável “i” com o valor inicial “0”.

Enquanto o valor de “i” for menor do que “4” faremos a seguinte tarefa: escreveremos o valor de “i” na tela e em seguida comamos 1 ao valor de “i”. Quando a variável for igual ou maior do que 4, o comando *while* para de realizar a tarefa.

Você consegue dizer o que o programa abaixo faz?

```
i = 1
while i <= 2 ** 32:
      print(i)
      i = i * 2
```

5.5. PRATICANDO COM O COMANDO WHILE

A seguir, temos uma lista de programas usando o comando *while*, sua função é analisar os programas, verificar se eles condizem com a função que lhe é atribuída. Caso esteja correto, use seu IDLE para verificar a saída. Caso esteja errado, corrija o programa e use o IDLE para verificar.

Programa 1:

```
#Escrever os números de 3 até 20
i = 3
while i < 20:
      print(i)
      i = i + 1
```

☐ Correto

☐ Incorret

Programa 2:

```
#Escrever 7 vezes a palavra “please”.
i = 0
while i < 8:
      print(please)
      i = i + 1
```

☐ Correto

☐ Incorreto

Programa 3:

```
#Fazer uma pergunta ao usuário, se a resposta for “s”, escrever na tela: “Você Venceu!”, se a resposta for “n” escrever: “Você perder1”, se for outra resposta (diferente de s ou n) fazer a pergunta novamente.

resposta = 0
while resposta != "s" and resposta != "n":
      resposta = input("Harry Potter é o melhor? [S/N]")
      if resposta.lower() == "s":
            print("Você Venceu!")
      elif resposta.lower() == "n":
            print("Você Perdeu!")
```

☐ Correto

☐ Incorreto

5.6. PROJETANDO

Estamos prontos agora para criar nosso primeiro jogo! Tudo bem, não vão ter gráficos ainda, mas não deixa de ser um jogo interessante, que você poderá depois mostrar pra seus familiares. O nome do jogo é:

ATRAVESSANDO O SERTÃO

Contexto:

Você é um justiceiro do cangaço, parente de Lampião, e encontrou um bando de ladrões que estavam vagando pelo sertão. Enquanto eles dormiam, você roubou a saca de dinheiro deles e fugiu no seu jumento rumo a cidade mais próxima, para devolver o dinheiro à polícia. Quando você estava a 20 metros de distância dos bandidos, eles notaram a perda e começaram uma perseguição.

Objetivo:

Atravessar o sertão (200 km), sem deixar seu jumento morrer de cansaço, morrer de sede ou ser alcançado pelos bandidos.

Pertences:

Além do jumento, você possui uma garrafa com água suficiente para 3 goles.
Obs: O jumento não necessitará de água durante o percurso.

Guia de programação:

- Crie um novo programa no IDLE, e euse o comando *print* para escrever um pequeno texto explicando o contexto do jogo. Use quantos comandos *print* você desejar para manter seu programa organizado.
- Em seguida crie 5 variáveis: kilômetros percorridos, nível de sede, cansaço do jumento, distância dos ladrões, goles de água. Defina as três primeiras variáveis com valor 0, a distância dos ladrões com valor de -20 e a quantidade de goles a seu critério (pode ser 3, 4, 5, etc).

- Em seguida você vai criar uma variável chamada *resposta* valor igual a 0. Esta variável será usada para manter ou finalizar o *Loop* que será criado no próximo passo.
- Crie um *Loop* usando o comando *while* onde a variável de controle é a variável *reposta*. Enquanto essa variável for diferente do valor 1, o *Loop* deverá permanecer funcionando. Quando ela for igual a 1, o jogo deverá ser finalizado.
- Dentro do *Loop* você deve criar o seguinte conjunto de comandos *print* referente as alternativas que o jogador poderá escolher:

```
17      print("")
18      print("A - Beba água da garrafa.")
19      print("B - Siga em frente em velocidade MODERADA.")
20      print("C - Siga em frente RAPIDO.")
21      print("D - Descansar")
22      print("E - Meus status.")
23      print("Q - Quit Game.")
24      print("")
```

- Após mostrar as opções na tela, peça ao jogador que escolha uma das opções. Guarde a informação dentro da variável *resposta*. Lembre que o jogador pode usr letra maíscula ou minúscula.
- Verifique a seguir se o jogador escolheu a opção *Q*, se sim, escreva na tela *"Fim de Jogo!"* e mude o valor da variável escrever para *"1"*, finalizando assim o *Loop*.
- Em seguida faça um comando *Elif* para verificar se a o jogador escolheu a letra *E.* Se esse for o caso, você deve escrever na tela os valores das variáveis correspondentes aos kilometros percorridos por você, a distância entre você e os bandidos e a quantidade de goles de água que lhe resta.
- Faça outro *Elif* para verificar se a escolha foi a letra *D*. Caso tenha sido este o caso você deve escrever uma frase avisando que o Jumento esta descansado. Deve ainda aumentar a distância percorrida pelos ladrões. Você não deve aumentar a sua kilometragem. Deve ainda fazer a variável "cansaço do Jumento" ser igual a "0".

  - Caso a alternativa escolhida seja "C", você deve aumentar a kilometragem em ate 15km (ou como você desejar), aumentar sua sede em 1, o cansaço do jumento em 2 (ou 3), e a distância dos bandidos deve ser também aumentanda.
  - Caso a alternativa escolhida seja a "B", deve-se aumentar sua velocidade em até 7 km, sua sede em 1, o cansaço do jumento em 1 e a distancia dos bandidos tbm deve ser aumentada.
- Caso seja a alternativa "A", você deve escrever na tela que você matou sua sede. Deve também diminuir em 1 a variável corrspondente a quantidade de goles e zerar a variável sede.
- Após verificar as alternativas escolhidas, deve-se então fazer um *IF* para verificar a sede do jogador. Se a variável sede for maior que 4 e enor que 6, deve emitir um aviso "Você esta com sede!". Se a variável sede for maior que 6, deve emitir a mensagem "Você morreu de sede!" e finalizar o jogo levando a variável *resposta* para 1.
- O mesmo deve ser feito com o cansaço do Jumento.
- Deve verificar tbm se a distância dos ladrões é igual ou superior sua kilometragem. Caso isso ocorra, siginifica que os ladrões lhe capturaram.

- Se o jogador conseguir atingir 200 kilometros sem morrer de sede, ser capturado ou o jumento morrer pelo cansaço, deve-se emitir a mensagem “Parabéns, você ganhou o Jogo!”.
- Obs: sinta-se livre para fazer qualquer modificação no jogo, seja em relação às regras, ao contexto e aos suprimentos. Use sua imaginação! Este é apenas um texto que servirá de base para você criar seu próprio jogo.

6. INTRODUÇÃO AOS GRÁFICOS

Agora que já temos um conhecimento básico dos principais comandos usados em *python*, daremos início aos nossos trabalhos com gráficos. No capítulo 6 iremos aprender a desenhar algumas formas geométricas bem como escrever texto em um quadro independente. Os conceitos por traz dos comandos não será o foco do capítulo, mas sim aprender a alterar, mudar cor e posicionar seus desenhos e texto, por isso, foque-se em aprender como modificar os comandos a principio; quando estiver mais familiarizados com estes, você poderá partir para os conceitos por traz de cada um dos comandos necessários para trabalhar as imagens.

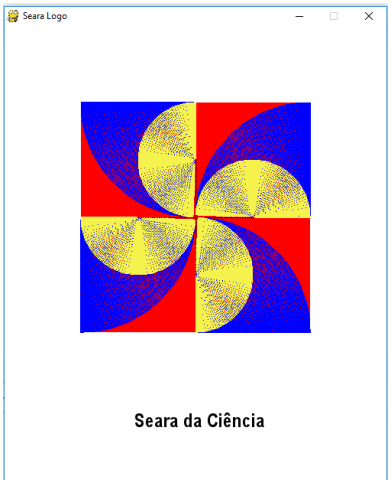


Figura 20 - Logo Seara da Ciência criado com comandos em python.

6.2. PYGAME

No início da programação as imagens não eram possíveis de serem criadas. Os programas eram por sua totalidade de cunho escrito e não estético com mostra a figura abaixo.

Figura 21 - Programa de computador de 1983. Fonte: http://www.hardware.com.br

Com o decorrer do tempo, novas linguagens de programação foram criadas a fim de proporcionar uma maior facilidade na obtenção de estruturas gráficas.

O python por si só não possui as ferramentas necessárias para se fazer um desenho. Por isso, precisamos antes de iniciar nossos rascunhos, trazer para o python essa biblioteca de ferramentas. Para isso, em sua primeira linha de comando, deve haver a seguinte requisição:

```
import pygame
```

6.3. CORES

A forma como o pygame trabalha com cores é bem simples (se comparado com algumas outras linguagens de programação). Ele usa o método RGB (*Red, Green, Blue*) para estabelecer padrões de cores.

O funcionamento é simples: As cores **Vermelho** (*Red*), **Verde** (*Green*) e **Azul** (*Blue*) possuem uma escala que varia de 0 a 255, onde 0 é a tonalidade mais próxima do Preto e 255 mais próxima do branco. Misturando essas três cores, podemos obter qualquer outra desejável com tonalidades diferente. Por exemplo:

(255,0,0) – 255 Red, 0 Green e 0 Azul

(0,255,0) - 0 Red, 255 Green e 0 Azul

(0,0,0) - 0 Red, 0 Green e 0 Azul

(255,255,255) - 255 Red, 255 Green e 255 Azul

Você pode descobrir como formar diversas cores usando o site http://www.colorpicker.com/

Para praticar vamos formar algumas cores no formato RGB?

Azul –
Rosa -
Cinza -
Amarelo -
Laranja -

Em nossos programas, será necessário definir variáveis que serão iguais ao formato RGB das cores. O pygame não possui essas cores pré-estabelecidas, então por isso é bom ter sempre em mente como formá-las e como declará-las.

6.4. UMA TELA PARA DESENHAR

Todo desenhista precisa a princípio de uma tela para pintar e desenhar. Essa tela pode ser branca, preta, azul, ou qualquer outra cor de sua preferência, e dentro dessa tela serão desenhados curvas, quadrados, círculos e letras.

Para fazer isso precisamos seguir alguns passos. O programa a seguir abre uma janela branca onde cada comando está sendo comentado com sua função:

```
import pygame # Importando biblioteca
pygame.init()# Iniciando pygame
```

```
BRANCO = (255, 255, 255) # Criando cor Branca

# Tamanho da tela
size = (400, 500)
# Criando tela com tamanho especificado
screen = pygame.display.set_mode(size)

# Título da tela, aparecerá na parte de cima.
pygame.display.set_caption("Tela em branco")

# Criação do Loop
Resposta = 0
clock = pygame.time.Clock()

while Resposta != 1: # Criação do Loop

      # Avaliar o que foi feito pelo usuário.
      for event in pygame.event.get():
      if event.type == pygame.QUIT: # Se apertar QUIT
            Resposta = 1 # Para sair do Loop

      screen.fill(WHITE) # Preenchendo tela com cor Branca

      #Essa área entre os dois comandos devem ficar os comandos de desenho.
      # Manter desenho na tela
      pygame.display.flip()
      # Loop será realizado 60 vezes por segundo
      clock.tick(60)

pygame.quit()# Finalizando pygame
```

Como foi dito antes, não se atente aos detalhes do comando de criação. O que você deve se preocupar no momento é em como fazer edições nesse programa. Para isso, vamos praticar um pouco: Tente modificar o tamanho do quadro, o título do quadro e a cor dele. Lembre-se de manter todos os comandos necessários para iniciar e fechar o pygame, bem como o Loop de desenho.

6.5. FORMAS GEOMÉTRICAS

Antes de desenhar qualquer coisa, devemos primeiro entender como o computador organiza a informação na tela.

Lembra da matemática, onde temos um plano cartesiano? O eixo horizontal chamado de abscissas e o horizontal de ordenadas? Observe a figura ao lado e tente lembrar- dos conceitos de matemática sobre os eixos e como achar os pontos.

O computador entenderá o quadro que você desenhou na seção 6.3 de forma similar ao plano cartesiano, no entanto, há uma

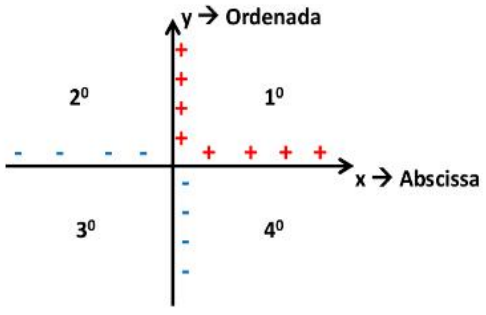


Figura 22 - Plano cartesiano. Fonte: www.estudopratico.com.br

diferença. Seu quadro será representado apenas pelo 4º quadrante do plano e ainda o eixo das ordenadas será invertido. Veja a imagem abaixo. Ela mostra a distribuição dos pontos na sua tela de desenho.

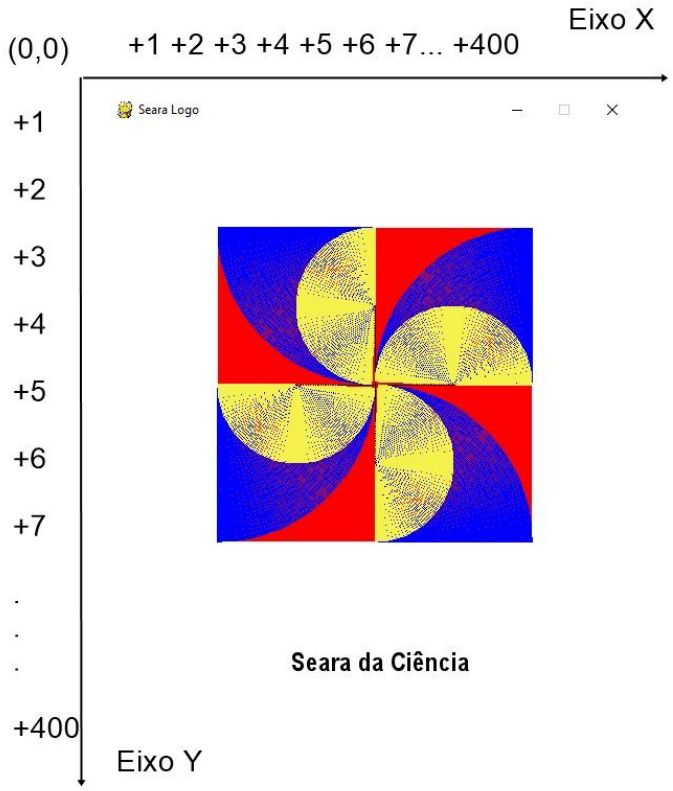


Figura 23 - Plano cartesiano pygame

Quando vamos desenhar um retângulo na tela por exemplo, devemos especificar os pontos onde ele se inicia, e o ponto em que ele termina. Isso acontece também com retas, polígonos em geral e texto. Abaixo veja como desenhar algumas formas geométricas e abaixo de cada exemplo uma breve explicação de como utiliza-las.

6.5.1. RETÂNGULOS

```
pygame.draw.rect(screen,COR,[Xi,Yi,Xc,Yc],espessura)
```

As partes que estão em verde devem ser modificadas.

COR – Deve-se colocar a variável correspondente a cor que você deseja pintar seu retângulo.

Xi e Yi – Ponto inicial do retângulo, correspondente ao canto superior esquerdo.

Xc e Yc – Comprimento e altura do retângulo

Espessura – Espessura do retângulo. Se colocar o valor 0 ele pinta toda a área do retângulo.

Exemplo:

```
1. pygame.draw.rect(screen,Vermelho,[100,100,100,100],0)
2. pygame.draw.rect(screen,Azul,[150,100,20,100],2)
```

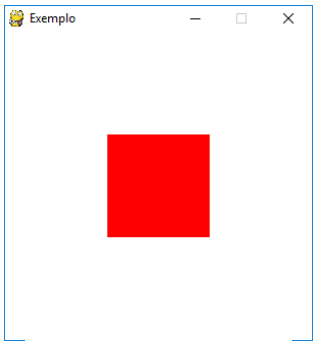


Figura 24 - Retângulo Vermelho de espessura 0.

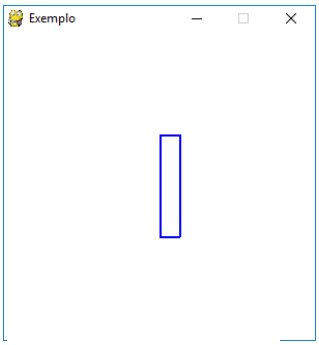


Figura 25 - Retângulo Azul de espessura 2.

Para praticar, desenhe vários quadrados e retângulos com diferentes cores, posições, comprimentos e alturas.

Importante saber que: todos os desenhos devem ficar entre os comandos *screen.fill() e pygame.display.flip().*

Os comandos de desenho devem estar alinhados com o *FOR*, ou seja, eles são um comando dentro apenas do *WHILE*

A ordem (cima para baixo) dentro do programa estabelece qual figura ficará sobre a outra caso haja sobreposição.

6.5.2. LINHAS

```
pygame.draw.line(screen, COR, [Xi, Yi], [Xf, Yf], Espessura)
```

De forma similar com o retângulo, pode-se mudar a COR e a espessura da linha.
Xi e Yi – Ponto inicial da linha
Xf e Yf – Ponto Final da Linha

6.5.3. ELIPSES

```
pygame.draw.ellipse(screen, COR, [Xi,Yi,Xc,Yc], Espessura)
```

A elipse caracteriza-se de forma semelhante ao retângulo em todos os parâmetros. Os pontos iniciais Xi e Yi correspondem ao canto superior esquerdo do retângulo que sobre-escreve a elipse de acordo com a figura abaixo:

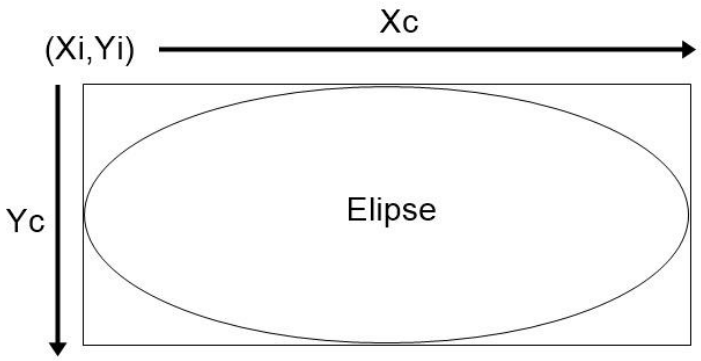


Figura 26 - Parâmetros para retângulo e elipse.

6.5.4. ARCOS

```
pygame.draw.arc(screen, COR, [Xi,Yi,Xc,Yc], Ai, Af, Espessura)
```

A diferença do arco para a elipse é que se deve informar o ângulo inicial (Ai) e o ângulo final (Af) do arco.

O *pygame* trabalha com ângulos em radianos, por isso, devem-se usar ângulos na forma de radianos como, por exemplo, PI, PI/2, (3*Pi)/2.
Os ângulos são dispostos no sentido anti-horário de uma circunferência.
A figura abaixo mostra a disposição dos ângulos.

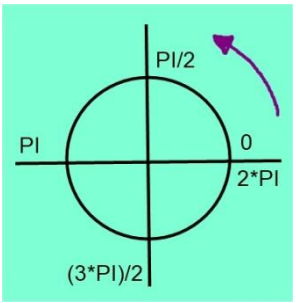


Figura 27 - Disposição dos ângulos em radianos.

6.5.5. TEXTO

A construção de texto é um pouco diferente das demais figuras geométricas. É necessário escrever algumas linhas de código antes de escrever o texto em si. Veja abaixo:

```
# Selecionar primeiro a fonte e o tamanho da letra.
font = pygame.font.SysFont('Calibri', 25, True, False)
text = font.render("Escreva texto aqui",True,COR)
# Posição inicial do texto
screen.blit(text, [Xi, Yi])
```

6.5.6. OUTROS POLÍGONOS

```
pygame.draw.polygon(screen, COR, [[X1,Y1], [X2,Y2], [X3,Y3]], Espessura)
```

De forma similar às outras formas geométricas, devemos determinar a cor e a espessura, e adicionar pontos que serão os vértices do polígono. O programa irá ligar os pontos na ordem que forem escritos e o último com o primeiro.

6.6. PROJETANDO

No projeto desse capítulo, faremos alguns desenhos para cenários de jogos, utilizando das figuras geométricas estudadas durante o capítulo e de um sistema de repetição por loop.
Veja as imagens abaixo e tente redesenha-las em seu computador usando linguagem *python* e a biblioteca *pygame.* Algumas dessas imagens possuem elementos repetidos. Tente usar um *Loop* para desenhar esses elementos ao invés de fazer várias linhas de comando.

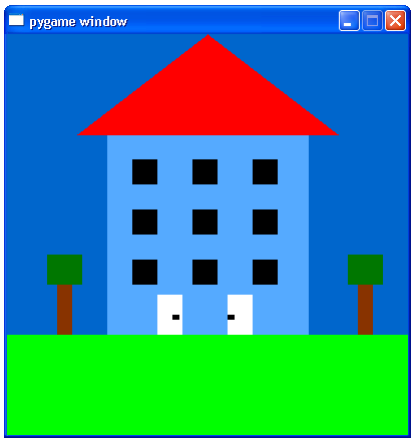


Figura 29 - Casa como cenário de fundo.

Figura 28 - Rampa de ski *como cenário de jogo.*

7. LISTAS E TUPLAS

Desde que começamos nossos trabalhos com *Python* já tivemos contato com alguns tipo de variáveis, dentre elas:

String – Conjunto de caracteres sem valor matemático, mais que carregam frases, nomes, e texto em geral;

Int – Abreviação de Inteiro e que corresponde aos números inteiros;

Float – Números flutuantes, ou de forma mais simples, números que não possuem valor inteiro.

No capítulo 7 vamos ter contato com mais dois tipos de variáveis que serão de extrema importância para o curso que se segue: *Listas* e *Tuplas*. Suas características, similaridades e diferenças, serão debatidas e exemplificadas nos próximos subitens.

7.2. DEFINIÇÕES: LISTAS E TUPLAS

Sabe quando você vai ao supermercado e leva uma lista com produtos que deve comprar? Na programação também podemos criar listas de informações, essas listas podem ser de dois tipos, mutáveis e não mutáveis.

Figura 30 - Lista. Fonte: noticias. universia.com.br

As listas mutáveis, chamadas apenas de **LISTAS** são aquelas que, depois de feitas, podem ser adicionados ou retirados valores, bem como substituí-los. Para criar uma *LISTA* em linguagem *Python*, basta declarar uma variável com valores, ou informações, dentro de colchetes, como no exemplo a seguir:

```
Idades = [101, 20, 10, 50, 60]
```

Com isso, acabamos de criar uma variável cujo nome é *Idades* e que possui como valor uma lista de números (101,20,10,50 e 60). O uso de colchetes define esta lista como mutável, e esses valores poderão ser substituídos depois. Veremos como fazer substituições nos próximos subitens.

Quando queremos criar uma lista que devem ser permanente, ou seja, imutável, basta substituir os colchetes por parênteses. Isso gerará uma **TUPLA**, que possui, assim como a lista, um conjunto de informações ou dados guardados em uma única variável. Veja o exemplo da criação de uma *tupla.*

```
Nomes = (“Carlos”, “André”, “Matheus”)
```

Observe que assim, como números, as *listas* e *tuplas* podem guardar strings.

7.3. ENDEREÇAMENTO

Antes de aprendermos a modificar as listas, é necessário saber como estas organizam seus elementos. Primeiro observe a figura abaixo:

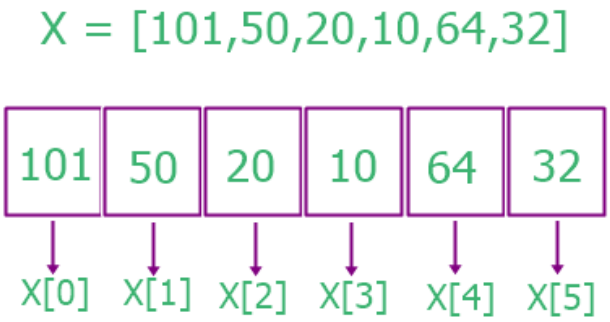


Figura 31 - Elementos de uma Lista

Na imagem podemos ver que foi declarada uma variável *X* do tipo *Lista* (devido ao uso de colchetes). A lista possui 6 elementos (101,50,20,10,64,32). Cada elemento desse irá ser guardado em uma caixa dentro da lista, e cada caixa possui um número de endereço. Esses endereços são números inteiros em ordem crescente que começam do *0*. Como a nossa lista *X* possui 6 elementos, os endereços serão: 0, 1, 2, 3, 4 e 5. Note que, o endereço é representado pelo nome da lista e o endereçamento entre colchetes. Com essa técnica, podemos agora descobrir quais os elementos de uma lista, buscando po seu endereço. Veja o exemplo abaixo:

```
x = [3,5,6,3,7]
print(x[0])
```

Resposta:
3

Observe que nossa lista tem apenas cinco elementos, logo, temos os endereços: x[0] = 3; x[1] = 5; x[2] = 6; x[3] = 3 e x[4] = 7. Caso você tente usar o elemento x[5], o programa enviará uma mensagem de erro. Esse mesmo recurso pode ser usado para *Tuplas.*

O método de endereçamento também funciona para variáveis do tipo *string*.

```
x = "Olá, meu nome é Harry!
print(x[2])
```

Resposta:
á

7.4. MODIFICANDO LISTAS

Como dito anteriormente, apenas as *Listas* podem ser modificadas, as *Tuplas* mantêm seus elementos os mesmos durante todo o processo. Para modificar um elemento de uma lista, basta direcionar um valor diretamente para o endereço desejado. Veja os exemplos:

Exemplo 1:

```
x = [3,5,6,3,7]
print(x)
x[2] = 3
print(x)
```

Resposta:
[3,5,6,3,7]
[3,5,3,3,7]

Exemplo 2:

```
x = ["Ovo", "Leite", "Queijo
print(x)
x[2] = "Feijão"
print(x)
```

Resposta:
["Ovo", "Leite", "Queijo"] ["Ovo", "Leite", "Feijão"]

Podemos ainda adicionar elementos à uma lista, sem precisar reescrever todos os elemtos. O comando usado é o *.append*.

Exemplo 3:

```
x = [“Ovo”, “Leite”, “Queijo”]
print(x)
x[2] = “Feijão”
print(x)
x.append(“Carne”)
```

Resposta:

```
[“Ovo”, “Leite”, “Queijo”]
[“Ovo”, “Leite”, “Feijão”]
[“Ovo”, “Leite”, “Feijão”, “Carne”]
```

Usando o comando *.append*, o novo elemento será adicionado a direita dos demais elementos.

Um outro comando bastante usado quando se trata de listas é o comando *len*. Este comando que vem do inglês *Length* (comprimento), mostra quantos elementos existem dentro da sua lista ou tupla.

Exemplo 4:

```
x = [“Ovo”, “Leite”, “Queijo”]
print(len(x))
```

Resposta:
3

7.5. PRATICANDO COM LISTAS

Antes de avançarmos nos conceitos de *Listas* e *Tuplas* vamos praticar o que aprendemos até aqui. Nos programas abaixo, mostre qual a saída que será obtida. Inicialmente, use apenas sua dedução lógica de programador. Depois, experimento copiar os programas no IDLE e ver a reposta que ele oferece. Compare e verifique suas respostas.

Programa 1:

```
Mercado = [“Maçã”, “Laranja”]
print(Mercado[0])
print(Mercado[1])
Mercado.append(2)
print(Mercado)
print(Mercado[2])
print(len(Mercado) * 4
```

Resposta:

Programa 2:

```
Mercado = (“Maçã”, “Laranja”)
print(Mercado[0])
```

```
print(Mercado[1])
Mercado.append(2)
print(Mercado)
```

Resposta:

7.6. USANDO LOOPS PARA MODIFICAR LISTAS

Loops são bastante usados para denotar valores à uma lista. Veja abaixo alguns exemplos de como eles podem ser usados.

```
lista = [101, 20, 10, 50, 60]
for item in lista:
      print(item)
```

Resposta:
101
20
10
50
60

A variável *item* irá assumir cada um dos valores da lista. Ao entrar no *loop FOR*, um comando *print* irá escrever cada um desses elementos.

```
lista = ["Colher","Garfo","Faca"]
for item in lista:
      print(item)
```

Resposta:
'Colher '
'Garfo'
'Faca'

O mesmo pode ser feito quando os elementos da lista são do tipo *string*.

```
lista = []
for i in range(4):
      user_input = input( "Digite um inteiro: ")
      user_input = int(user_input)
      lista.append(user_input)
      print(lista)
```

A princípio criou-se uma lista vazia. Dentro do *Loop* é pedido que o usuário digite um número inteiro com o comando *input.* Esse valor é então convertido para um número inteiro com o comando *int*. A cada novo *loop*, o valor digitado pelo usuário é acrescentado na lista com o comando *.append*. No final de cada *Loop* é escrito na tela seuselementos com o comando *print*. O *loop* irá repetir o procedimento quatro vezes.

Resposta:
Digite um inteiro: 4
[4]
Digite um inteiro: 5
[4, 5]
Digite um inteiro: 2
[4, 5, 2]
Digite um inteiro: 1
[4, 5, 2, 1]

7.7. PRATICANDO COM LOOPS E LISTAS

Vamos agora praticar com *loops.* Desenvolva os programas propostos abaixo. Lembre-se das regras delimitadas anteriomente para uso de *listas* e *tuplas*.

Programa 1:

Faça um programa que crie uma lista vazia a princípio. Em seguida, usando um *loop FOR*, peça ao usuário que escreva itens que deseja comprar no mercado. Adicione essas mercadorias na lista criada e imprima os itens da lista após o usuário indicar 5 elementos.

Programa 2:

Aprimore o seu programa anterior! Adicione um conjunto que adicone na lista infinitas quantidades de elementos. A *loop* deve ser finalizado apenas quando o usuário escrever a palavra "QUIT".

Dica: Você pode usar um comando *IF* ou um *loop WHILE* para essa tarefa.

7.8. UNICODE

O computador no qual você salva e executa seu programa não entende letras. Isso mesmo! Tudo que você escreve no IDLE é convertido pelo executador/copilador para um conjunto de números.

O **UNICODE** é um sistema de codificação/decodificação. Ele serve para associar as letras do alfabeto que conhecemos para um determinado número entre 0-127.

Abaixo você encontra uma tabela com os valores em UNICODE de alguns caracteres que usamos diariamente. A coluna *Cha* mostra o caracter, enquanto a coluna *ORD* mostra o valor em UNICODE. Veja que alguns espaços estão sem o valor *ORD.* Ache esses valores usando o comando:

ORD("Caractere")

Onde você deve substituir o nome *Caractere* pelo desejado.

Tabela 5- Valores UNICODE de alguns caracteres.

Cha	ORD	Cha	ORD	Cha	ORD	Cha	ORD
0	48	A	65	K	75	U	85
1	49	B	66	L	76	V	86
2	50	C	67	M	77	W	87
3	51	D	68	N	78	X	88
4	52	E	69	O	79	Y	89
5	53	F	70	P	80	Z	90
6	54	G	71	Q	81		
7	55	H	72	R	82		
8	56	I	73	S	83		
9	57	J	74	T	84		

Cha	ORD	Cha	ORD	Cha	ORD	Cha	ORD	Cha	ORD
a	97	k	107	u	117	.		Â	
b	98	l	108	v	118	*		Â	
c	99	m	109	w	119	-		Ã	

d	100	n	110	x	120	/		ã	
e	101	o	111			|		(	
f	102	p	112	?		\		)	
g	103	q	113	!		:		{	
h	104	r	114	@		[		}	
i	105	s	115	#		]		=	
j	106	t	116	;		Ç		%	

Você pode usar o comando **CHR** para fazer o reverso: mudar de UNICODE para caracteres.

Agora veja algumas aplicações do uso do UNICODE.

Exemplo 1:

```
frase = input("Digite uma frase: ")

for i in
range(len(frase)):
      x = ord(frase[i])
      x +=1
      y = chr(x)
      print(y, end="")
```

```
Digite uma frase: Oi tudo bom?
.
.
```

7.9. PROJETANDO

Vamos usar um pouco das técnicas qe aprendemos sobre *Listas* para criar mais um joguinho. A imagem a seguir mostra o exemplo de uma casa onde o jogo irá acontecer. Por enquanto ainda não vamos usar imagens e animações, apenas texto. Mas o contexto pode ficar sob sua imaginação.

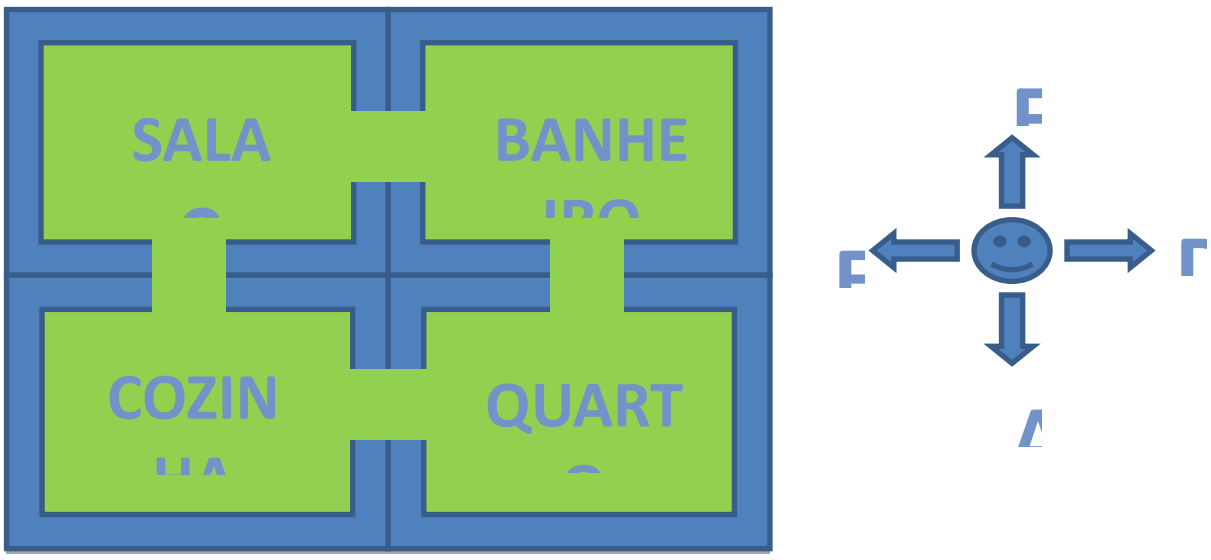


Figura 32 - Representação do espaço onde o jogo irá acontecer,

Contexto:

O contexto do jogo fica a cargo do programador.

Objetivos:

O jogo se passa em um ambiente desconhecido pelo jogador. Ele inicia a partida em um dos cômodos e seu objetivo é chegar em um outro determinado cômodo. O jogador pode sempre escolher as opções Frente (F), Direita (D), Esquerda (E) e Atrás (A).

Como fazer:

Crie uma lista vazia chamada *lista_quartos*.

Crie uma variável chamada *room*. Essa variável será uma lista formada por:

1º elemento: Uma *string* com o nome de um cômodo;

2º elemento: O número do cômodo à direita;

3º elemento: O número do cômodo acima;

4º elemento: O número do cômodo à esquerda;

5º elemento: O número do cômodo abaixo.

Após criar a variável para o primeiro cômodo, adicione o mesmo na *lista_quartos* usando o comando *.append*.

Faça o mesmo para os outros cômodos.

Crie uma variável com o nome *quarto_atual* e defina com o valor 0, correspondente à sala. Ou seja, o jogador irá iniciar a partida na sala.

Crie um comando *While* que escreva o contexto da história e mostre as opções que o jogador possa escolher. Salve a opção do jogador em uma variável *escolha*.

Faça um conjunto de comandos *IF* e *ELIF* que associe a resposta do jogador com uma letra “d”, “f”, “e” ou “a”. Para cada escolha o *quarto_atual* deve mudar seu valor para o novo cômodo em que ele se encontra.

Caso o jogador atinja o cômodo final (Quarto - 3) emita um “Fim de Jogo”

8. INTRODUÇÃO À ANIMAÇÃO

Até agora aprendemos apenas a fazer alguns joguinhos simples sem uso de imagens. Neste capítulo vamos dar uma pequena introdução em como fazer animações simples. Para isso, é bastante importante que você tenha em mente o programa base para desenhar imagens na tela, esse assunto foi estudado no capítulo 6 dessa apostila. Além do capítulo 6, o uso de *Loop* através de comandos *While* e *For* é de extrema necessidade. Por isso, aconselhamos você revisar esses capítulos e tirar todas as dúvidas presente.

8.2. UM QUADRADO QUE NÃO PARA

Nossa primeira animação diz respeito a um quadrado em um fundo preto (ou qualquer outra cor desejada) que irá se mover dentro da tela e rebater ao chegar no limite da tela. A figura abaixo mostra um esquema da animação.

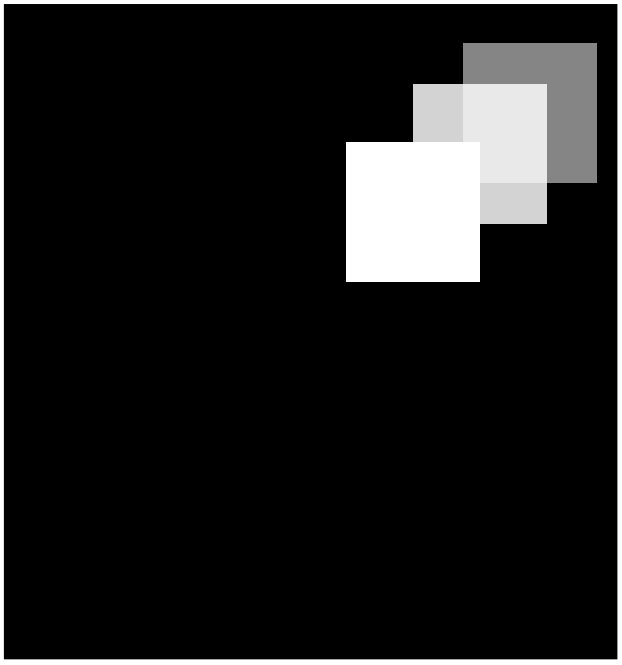


Figura 33 - Esquema da animação proposta.

Para fazer essa animação, deve-se recorrer ao programa base de animação do capítulo 6. Copie e cole o programa base no seu IDLE. Faremos as seguintes modificações:

- Defina as cores branco e preto;
- Defina o tamanho da tela em *size(400,600);*
- Defina duas variáveis chamadas *dim_x* e *dim_y.* Elas serão responsáveis por mostrar a posição inicial do quadro, onde ele irá começar. Escolha qualquer valor que esteja dentro do quadro preto (400,600);
- Defina mais duas variáveis chamadas *dim_x_mud* e *dim_y_mud*. Essas variáveis serão responsáveis por movimentar o quadrado.
  Escolha valores pequenos para elas (entre 1 e 10). Quanto maior a variável, mais rápido o quadro irá se movimentar.
- Dentro do *Loop*, preencha a tela com a cor preta.

- Desenhe um retângulo de cor branco. A posição inicial do quadrado se dá com as variáveis *dim_x* e *dim_y.* O tamanho do quadrado fica a seu critério.
- Para fazer o quadradro se mover, precisamos usar algo simples: Adicionar um valor tanto na posição X como na posição Y inicial do quadrado. Assim, a cada L*oop*, o quadrado começara numa posição nova. Para isso, vamos dizer então que *dim_x* e *dim_y* será igual a ele mesmo mais a variável *dim_x_mud* e *dim_y_mud* que foi declarada no início do programa:
-

```
dim_x += dim_x_mud
dim_y += dim_y_mud
```

- Agora, se o quadradro atingir o limite da tela, o que devemos fazer? Essa parte recisa ser pensada com detalhe e clareza. Veja um exemplo de como fazer logo abaixo; tente descrever como essas linhas de comando irão funcionar:
-

```
if dim_x > 350 or dim_x < 0 :
dim_x_mud *= -1
if dim_y > 550 or dim_y < 0 :
dim_y_mud *= -1
```

- Tente seguir esses passos para montar o quadrado que se move. Caso tenha dúvidas, no funcionamento do programa, pergunte ao monitor como deve ser feito com mais clareza.

8.3. PROJETANDO

A seguir, temos um código de um programa simples que fará nevar em sua tela. Observe cada linha de comando e veja como o programa funciona.

O projeto para esse capítulo 8 será a investigação do programa. Uma-se com mais dois colegas e tentem decifrar como o programa funciona. Vocês terão que mostrar os resultados da análise para o restante da classe. Tente modificar o programa (cor de fundo, cor da neve, tamanho da neve, velocidade que ela cai) para customiza-lo.

Importante saber que, muitas vezes o trabalho de um programador esta em entender e analizar códigos!

FAZENDO NEVAR

```
import pygame
import random

pygame.init()
preto = [0, 0, 0]
branco = [255, 255, 255]

SIZE = [400, 400]
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption("Fazendo Nevar")

lista_neve = []

for i in range(50):
    x = random.randrange(0, 400)
    y = random.randrange(0, 400)
    lista_neve.append([x, y])

clock = pygame.time.Clock()
resposta = 0

while resposta!=1:
    for event in pygame.event.get():
            if event.type == pygame.QUIT:
                    resposta = 1


    screen.fill(preto)

    for i in range(len(lista_neve)):

            pygame.draw.circle(screen, branco, lista_neve[i], 2)
            lista_neve[i][1] += 1

            if lista_neve[i][1] > 400:

                    y = random.randrange(-50, -10)
                    lista_neve[i][1] = y
                    x = random.randrange(0, 400)
                    lista_neve[i][0] = x

    pygame.display.flip()
    clock.tick(20)

pygame.quit()
```

Considerações:

- A biblioteca *random* foi adicionada para que fosse possível usar o comando *random.randrange(A,B)*. Esse comando gera um número aleatório que esteha entre A e B.
- O comando que desenha circunferências é similar aos usados para desenhar retângulos e arcos. Defini-se a cor, e a posição inicial do quadrado que engloba o círculo.

9. FUNÇÕES

Funções nada mais é do que um recurso da linguagem python que irá ajudar a organizar seu programa, tornan-do-o mais simples e fácil de ser entendido.

9.2. DEFININDO UMA FUNÇÃO

Observe a imagem a seguir para que possamos tirar algumas conclusões sobre o funcionamento desse recurso.

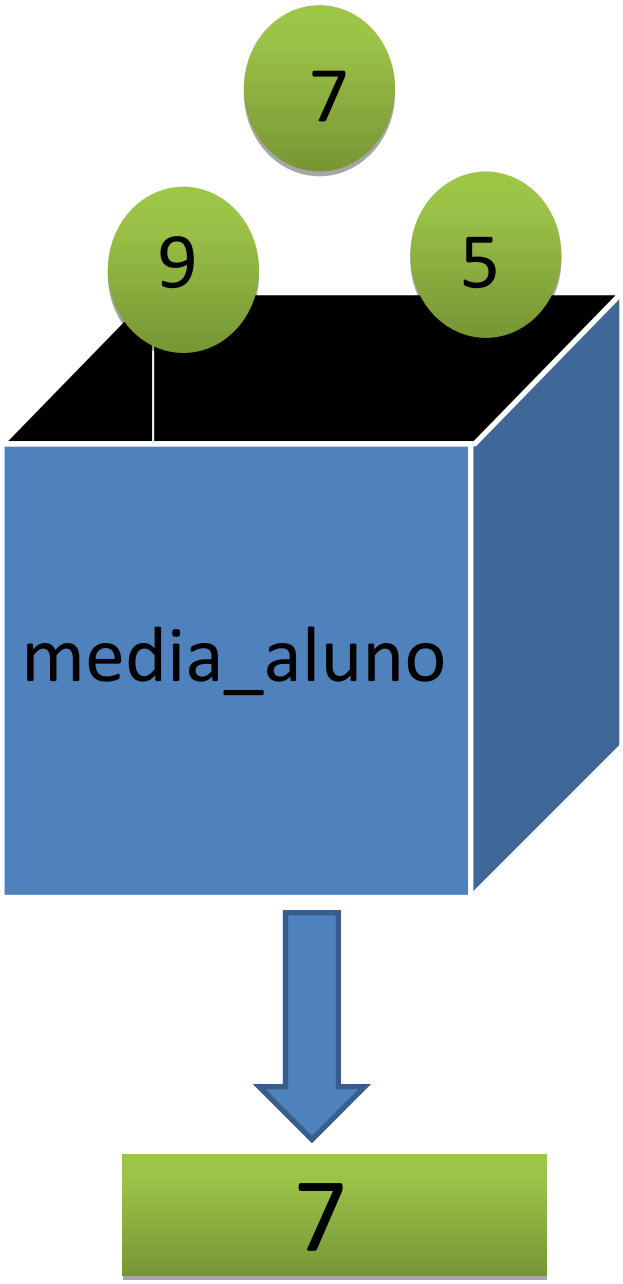


Figura 34 - Exemplo de uma função

A função funciona como uma caixa de comandos. Dentro dela você pode criar qualquer sequencia de códigos da sua preferência. No exemplo da imagem, temos uma caixa que recebe 3 valores (notas de um aluno) e libera uma resposta (a média desse aluno). A associção com uma caixa se dar pelo fato de, o programa principal não vê o que a função irá fazer, apenas lhe

dará valores e irá receber outro(os) em troca.

Mas, assim como as variáveis, as funções tem que ser declaradas. Vejamos um exemplo de como fazer uma função:

```
def media_aluno(nota1, nota2, nota3):
    media = (nota1+nota2+nota3)/3
    print("A média do aluno é: ", media)
    return media
```

- Veja que, devemos iniciar uma função com *def*. Isso indicará ao computador que as linhas (com espaço) abaixo fazem parte de sua função.
- Após *def* define-se o nome da função (mesma regra ao se dar nomes para variáveis).
- Em seguida, usa-se parenteses *"()"* e dois pontos *":"*.
- Dentro dos parênteses pode haver (ou não) as variáveis de entrada.
- As linhas de código abaixo da função farão parte da função, se estiverem com espaçamento.
- No final da função pode haver (ou não) o comando *return* que irá retornar o valor de alguma variável criada dentro da função.

Algumas considereações importantes sobre as funções:
Variáveis criadas dentro das funções não podem ser usadas fora da mesma.
Pode-se criar variáveis dentro da função com mesmo nome que variáveis fora dela; no entanto, deve-se entender que elas são vistas de forma diferenciada pelo computador. Modificando a variável local (0dentro da função) não modificará a variável fora da função (ou geral).

9.3. CHAMANDO UMA FUNÇÃO

Declarar uma função é simples, vimos que basta usar o comando
def nome_da_função (variáveis de entrada):
Mas agora, como usamos as funções?

Simples! Basta, durante o programa principal (ou dentro de uma outra função) escrever o nome da função desejada seguida de ():. Dentro dos parênteses, deve-se colocar os valores para variáveis de entrada. Veja um exemplo:

```
def f():
    x = 22
    print(x)

f()
```

No exemplo acima, declaramos uma função chamada *f*.
A função *f* não possui variáveis de entrada.

Dentro da função temos apenas o comando *x = 22* e *print(x).* O programa só irá escrever na tela se a função for chamada em algum outro momento do programa principal.

Em seguida, durante o programa principal, chamamos a função com a linha *f().*

Ao fazer isso, chamamos a função *f* que irá atribuir o valor 22 à variável *x* e irá escrever na tela esse valor.

Abaixo, temos alguns códigos usando funções. Novamente, você como programador tem que verificar se o que cada código realizará. Após fazer suas observações, simule cada um dos códigos no seu IDLE e verifique suas respostas.

9.4. EXEMPLOS A SEREM EXAMINADOS

Example 1:

```
def a():
      print("A")
def b():
      print("B")
def c():
      print("C")
a()
```

Example 2:

```
def a():
      b()
      print("A")
def b():
      c()
      print("B")
def c():
      print("C")
a()
```

Example 3:

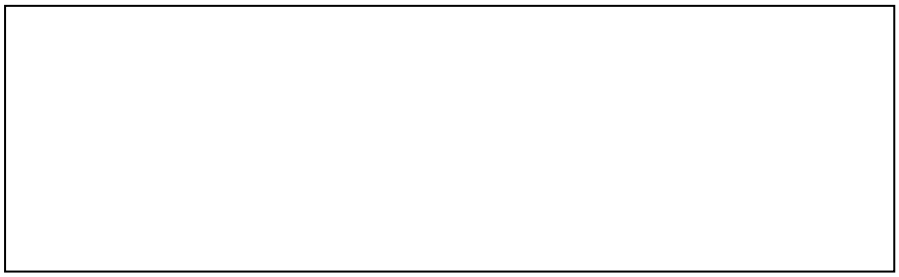
```
def a():
      print("A")
      b()
def b():
      print("B")
      c()
def c():
      print("C")
a()
```

Example 4:

```
def a():
      print("A inicio")
      b()
      print("A fim")
def b():
      print("B inicio")
      c()
      print("B fim")
def c():
      print("C inicio e fim")
a()
```

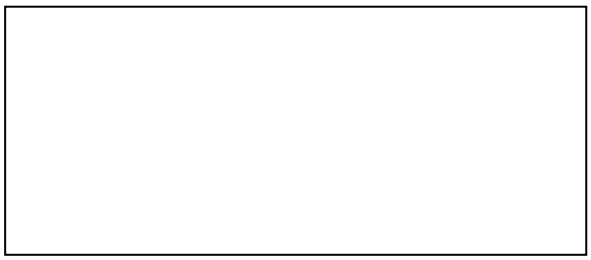
Example 5:

```
def a(x):
      print("A inicio, x =",x)
      b(x + 1)
      print("A fim, x =",x)
def b(x):
      print("B inicio, x =",x)
      c(x + 1)
      print("B fim, x =",x)
def c(x):
      print("C inicio e fim, x =",x)
a(5)
```



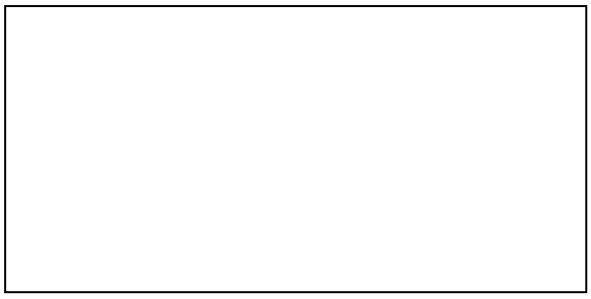
Example 6:

```
def a(x):
      x = x + 1
x = 3
a(x)
print(x)
```



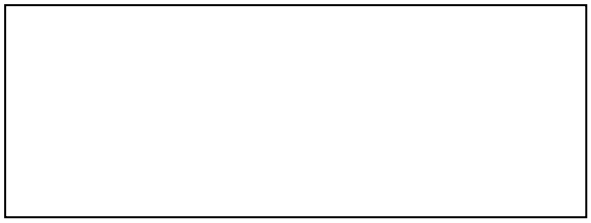
Exemplo 7:

```
def a(x):
      x = x + 1
      return x
x = 3
a(x)
print(x)
```

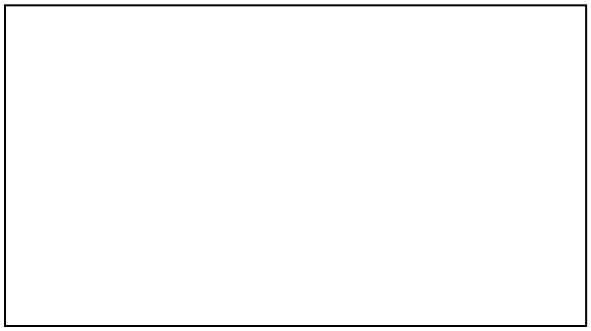


Exemplo 8:

```
def a(x):
      x = x + 1
      return x
x = 3
x = a(x)
print(x)
```



Exemplo 9:



```
def a(x, y):
      x = x + 1
      y = y + 1
      print(x, y)
x = 10
y = 20
a(y, x)
```

Exemplo 10:

```
def a(x, y):
      x = x + 1
      y = y + 1
      return x
      return y
x = 10
y = 20
z = a(x, y)
print(z)
```

Exemplo 11:

```
def a(x, y):
      x = x + 1
      y = y + 1
      return x, y
x = 10
y = 20
z = a(x, y)
print(z)
```

Exemplo 12:

```
def a(x, y):
      x = x + 1
y = y + 1
      return x, y
      return x, y
x = 10
y = 20
x2, y2 = a(x, y)
print(x2)
print(y2)
```

Vamos trabalhar um pouco mais as funções.
A seguir temos um pequeno teste a ser feito. O teste consiste de algumas questões que valem determinado ponto. Vamos ver quantos pontos você consegue?

1) (5 pts) Crie uma função chamada *min3* que recebe três valores e mostre o menor valor entre os três. Você deve usar o comando *return* e não o comando *print* dentro da função. O coando *print* só deve ser usado fora da função. Tente usar os comando *if/elif/else* para descobrir qual o menor núemero.

   Após fazer o programa, teste os seguinte parâmetros:

```
print(min3(4, 7, 5))
print(min3(4, 5, 5))
print(min3(4, 4, 4))
print(min3(-2, -6, -100))
print(min3("Z", "B", "A"))
```

2) (5 pts) Crie uma função chamada *box* que recebe uma altura e uma largura. A função irá então dsenhar um quadrado feito de asteriscos (*) com a altura e largura dada à função. Uma dica é usar dois comando *FOR,* um dentro do outor. Após terminar a sua função, teste os seguintes comando:

```
box(7,5)
print()
box(3,2)
print()
box(3,10)
```

3) (5 pts) Crie uma função chamada *find* que procura um valor dado (chamado *Key*) dentro de uma lista de números. O programa deve imprimir a posição, dentro da lista, em quea chave foi encontrada.
   Por exemplo, no código abaixo, o programa deve imprimir: 11,13 e 5.

```
my_list = [36, 31, 79, 96, 36, 91, 77, 33, 19, 3, 34, 12, 70, 12, 54, 98, 86, 11, 17, 17]

find(my_list, 12)
find(my_list, 91)
find(my_list,  80)
```

4) (5 pts)Crie uma função que receba um número (n)e retorne uma lista com com *n* elementos. Esses elementos devem ser números aleatórios entre 1 e 6. Para essa tarefa, use a biblioteca *random* e o comando *random.randrange(1,7)*.
5) (5 pts)Faça uma função que receba uma lista com vários números e receba também um *key* número. A função deve retornar a quantidade de vezes que o *key* número aparece na lista. Por exemplo, no comando abaixo, deverá escrever na tela “3”, porque o número 4 aparece três vezes na lista.

```
count = count_list([1,2,3,4,4,4,2,1],4)
print(count)
```

6) (5 pts) Crie uma função que receba uma lista de valores e calcule a média desses valores. Obs: A função deve ser feita para receber qualquer lista com quantidades indefinidas de elementos.

Total de pontos:_____

10. CONTROLES

Figura 35 - Exemplo de console *para vídeo games.*

Nesse capítulo 10 vamos dar início aos códigos que permitem o controle de elementos visuais através do mouse ou do teclado. Para isso, voltaremos a usar o programa base para gráficos e conceitos de funções. Após esse capítulo, você estará apto a fazer diversos tipos de jogos simples. Então, tente entender como funciona corretamente o controle na linguagem *Python* e se divirta.

10.2. USANDO O MOUSE

O controle através do mouse é muito simples, mais simples que pelo teclado, basta adicionar uma simples função e esta lhe dará uma *TUPLA* com dois valores: o primeiro é a posição *X* onde o mouse se encontra e o segundo valor é a posição *Y*.

Figura 36 - Exemplo de Mouse.

A função é descrita a seguir:

pygame.mouse.get_pos()

Vamos fazer junto um exemplo:

1) Crie uma função que desenhe um boneco simples igual o mostrado a seguir. O detalhe da função é que, nas coordenadas onde serão desenhados cabeça, corpo, braços e pernas, deve-se haver uma variável *X* e *Y* que determinará a origem dos desenhos. Veja o exemplo de como fazer a cabeça do boneco.

```
def boneco(screen, x,y):
    #Cabeça
    pygame.draw.ellipse(screen, ROXO, [x,y,10,10], 0)
```

2) Copie a função criada no programa base para gráficos. Ela deve se localizar logo após o comando *import pygame*. Lembre-se de adicionar todas as cores necessárias e de pintar a tela de branco.

3) Após as modificações necessárias, chame a função *boneco* dentro da área destinada para desenhos. Ao chamar a função, as variáveis de entrada da função devem ser dadas através do comando do mouse. Veja o exemplo:

```
# Comando para pegar Tupla do mouse (x,y)
pos = pygame.mouse.get_pos()
x = pos[0]
y = pos[1]
# Drawing section
boneco(screen, x, y)
```

4) Observe que o simbolo do mouse fica sendo mostrado na tela.Você pode tirar o símbolo usando o comando:

```
pygame.mouse.set_visible(False)
```

10.3. USANDO TECLADO

Figura 37 - Teclado

O uso do teclado é um pouco mais complexo do que o mouse. Para trabalhar com o teclado é necessário estudar os eventos que ocorrem durante o *Loop* principal.

Você deve ter notado que, para fechar a tela de animação, o programa base possui um comando do tipo:

```
for event in pygame.event.get():
      if event.type == pygame.QUIT:
            reposta = 1
```

Esse For que é usado serve para detectar eventos que ocorram. Eventos podem ser descritos como a interferência do "jogador" no jogo. Quando apertamos uma tecla do teclado para fazer o elemento se movimentar na tela, ou quando apertamos algum botão do mouse são exemplos dessa interferência.

O comando acima cria um *IF* que verifica se o evento que ocorreu foi o jogador querendo fechar a tela. Para verificar outros eventos (apertar a tecla para cima, para baixo, direita, esquerda, botão esquerdo do mouse) devemos usar um comando *IF/Elif/Else* para cada necessidade.

Dois comandos importantes são KEYDOWN e KEYUP usados para quando precionamos e soltamos uma tecla respectivamente. Abaixo você pode ver uma tabela com várias outras funções, com sua correspondente tecla do teclado.

Código Pygame	Comando
K_BACKSPACE	backspace
K_SPACE	space
K_DELETE	delete
K_LEFT	Seta Esquerda
K_RIGHT	Seta Direita
K_UP	Seta Cima
K_DOWN	Seta Baixo
K_0	0

K_1	1
K_a	a
K_z	z
K_KP_DIVIDE	/
K_KP_MULTIPLY	*
K_KP_MINUS	-
K_KP_PLUS	+

Veja a seguir um exemplo que faz o boneco se movimentar na tela usando o teclado. Observe o programa e tente entender o funcionamento.

```
import pygame

def boneco(screen, x,y):
    # Cabeça
    pygame.draw.ellipse(screen, BLACK, [x,y,10,10], 0)
    #Pernas
    pygame.draw.line(screen, BLACK,[5+x,17+y], [10+x,27+y], 2)
    pygame.draw.line(screen, BLACK, [5+x,17+y], [x,27+y], 2)
    #Corpo
    pygame.draw.line(screen, RED, [5+x,17+y], [5+x,7+y], 2)
    # Braços
    pygame.draw.line(screen, RED, [5+x,7+y], [9+x,17+y], 2)
    pygame.draw.line(screen, RED, [5+x,7+y], [1+x,17+y], 2)

pygame.init()

BLACK = [0, 0, 0]
WHITE = [255, 255, 255]
RED = [255,0,0]

size = [400, 500]
screen = pygame.display.set_mode(size)

# velocidade de movimento
x_speed = 0
y_speed = 0
# Posição atual
x_coord = 10
y_coord = 10

resposta = 0
clock = pygame.time.Clock()

pygame.mouse.set_visible(False)

while resposta!=1:

    screen.fill(WHITE)
    for event in pygame.event.get():

            if event.type == pygame.QUIT:
                    resposta = 1

            elif event.type == pygame.KEYDOWN:
```

```
45                  if event.key == pygame.K_LEFT:
46                        x_speed = -3
47                  elif event.key == pygame.K_RIGHT:
48                        x_speed = 3
49                   elif event.key == pygame.K_UP:
50                        y_speed = -3
51                  elif event.key == pygame.K_DOWN:
52                        y_speed = 3
53
54            elif event.type == pygame.KEYUP:
55                  if event.key == pygame.K_LEFT or event.key ==
pygame.K_RIGHT:
56                        x_speed = 0
57                  elif event.key == pygame.K_UP or event.key ==
pygame.K_DOWN:
58                        y_speed = 0
59
60     x_coord += x_speed
61      y_coord += y_speed
62
63     man(screen, x_coord, y_coord)
64     pygame.display.flip()
65     clock.tick(60)
66
67 pygame.quit()
```

Teste o programa no seu IDLE e veja o que ocorre. Tente modificar as cores do boneco, bem como seu formato e a tela de fundo.

10.4. PROJETANDO

Agora é com você!
No nosso ultimo projeto, você, novo programador, deve criar seu próprio *Game*. Use o conhecimento adiquirido até agora para montar seu jogo! Crie cenários, personagens, use a imaginação. No final, você deverá apresentar seu jogo para a turma. Então divirta-se e aplique seus conhecimentos.

11. LISTA DE COMANDOS PRINCIPAIS

import – Adiciona uma biblioteca no programa

print() – Escreve na tela o argumento entre parênteses, podendo ser "strings" ou valores inteiros, float, etc.

input() – Recebe informação do usuário. A informação é recebida em forma de string, logo, devem-se usar métodos de conversão para outros tipos de objeto.

if/elif/else – Estrutura Condicional. O if possui maior grau de decisão, seguido pelos elif (que podem ser mais de um em uma mesma condição) e o else usado para condições que não possuem argumento.

.lower() – Condiciona os caracteres armazenados em uma determina variável para caracteres minúsculos. Esse comando não modifica o conteúdo da variável.

for – Estrutura de repetição/Loop. Usado quando se tem de forma definida a quantidade de interações a serem realizadas.

while – Estrutura de repetição/Loop. Usada quando não se tem a quantidade exata de repetições, mas o limite (quando se deve parar).

Random.Randrange() – Necessita da biblioteca Random. Gera números aleatórios dentro dos limites especificados como parâmetros.

Random.random() – Necessita biblioteca Random. Gera um número aleatório entre 0 e 1 (float número).