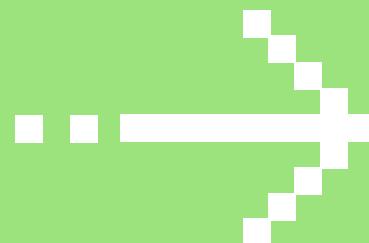


# INTRODUÇÃO À PROGRAMAÇÃO E ESTRUTURAS DE DADOS FUNDAMENTAIS COM PYTHON PARA PROGRAMADORES C



Antônio Rodrigo Delepiane De Vit  
Sidnei Renato Silveira



Compartilhando conhecimento

# INTRODUÇÃO À PROGRAMAÇÃO E ESTRUTURAS DE DADOS FUNDAMENTAIS COM PYTHON PARA PROGRAMADORES C

DOI: doi.org/10.36599/editpa-ipedfp



Antônio Rodrigo Delepiane De Vit  
Sidnei Renato Silveira



Compartilhando conhecimento

INTRODUÇÃO À PROGRAMAÇÃO E  
ESTRUTURAS DE DADOS FUNDAMENTAIS COM PYTHON PARA PROGRAMADORES C

**Editor Chefe**

Dr Washington Moreira Cavalcanti

**Autores**

Antônio Rodrigo Delepiane De Vit  
Sidnei Renato Silveira

**Conselho Editorial**

Dr. Lais Brito Cangussu

Dr. Rômulo Maziero

Msc Jorge dos Santos Mariano

Dr Jean Canestri

Msc Daniela Aparecida de Faria

Dr Paulo Henrique Nogueira da Fonseca

Dr Marcos Pereira dos Santos

Msc Edgard Gonçalves da Costa

**Projeto Gráfico e Diagramação**

Departamento de arte Synapse Editora

**Editoria de Arte**

Maria Aparecida Fernandes

**Revisão**

Os Autores

2023 by Synapse Editora

Copyright © Synapse Editora

Copyright do Texto © 2023 Os autores

Copyright da Edição © 2023 Synapse Editora

Direitos para esta edição cedidos à  
Synapse Editora pelos autores.

Todo o texto bem como seus elementos, metodologia, dados apurados e a correção são de inteira responsabilidade dos autores. Estes textos não representam de forma alusiva ou efetiva a posição oficial da Synapse Editora.

A Synapse Editora não se responsabiliza por eventuais mudanças ocorridas nos endereços convencionais ou eletrônicos citados nesta obra.

Os livros editados pela Synapse Editora, por serem de acesso livre, *Open Access*, é autorizado o download da obra, bem como o seu compartilhamento, respeitando que sejam referenciados os créditos autorais. Não é permitido que a obra seja alterada de nenhuma forma ou usada para fins comerciais.

O Conselho Editorial e pareceristas convidados analisaram previamente todos os manuscritos que foram submetidos à avaliação pelos autores, tendo sido aprovados para a publicação.



Compartilhando conhecimento

2023

INTRODUÇÃO À PROGRAMAÇÃO E  
ESTRUTURAS DE DADOS FUNDAMENTAIS COM PYTHON PARA PROGRAMADORES C

D278i De Vit, Antônio Rodrigo Delepiane

Introdução à programação e estruturas de dados fundamentais com Python para programadores C.

Autores: Antônio Rodrigo Delepiane De Vit; Sidnei Renato Silveira  
Belo Horizonte, MG: Synapse Editora, 2023, 61 p.

Formato: PDF

Modo de acesso: World Wide Web

Inclui bibliografia

ISBN: 978-65-88890-30-1

DOI: [doi.org/10.36599/editpa-ipedfp](https://doi.org/10.36599/editpa-ipedfp)

1. Python 2. Linguagem C de programação, 3. Computação,  
4. Análise de sistemas.

I. Introdução à programação e estruturas de dados fundamentais com Python para programadores C.

II. Antônio Rodrigo Delepiane De Vit  
Sidnei Renato Silveira

CDD: 005 - 005.10218

CDU: 004 - 004.43

**SYNAPSE EDITORA**

Belo Horizonte – Minas Gerais

CNPJ: 20.874.438/0001-06

Tel: + 55 31 98264-1586

[www.editorasynapse.org](http://www.editorasynapse.org)

[editorasynapse@gmail.com](mailto:editorasynapse@gmail.com)



Compartilhando conhecimento

2023

# PREFÁCIO

Este livro apresenta os conceitos de programação de computadores e estruturas de dados fundamentais, utilizando as linguagens de programação C e Python. A intenção desta obra é de que a mesma sirva como material didático para as disciplinas introdutórias de programação e estruturas de dados para alunos que não possuem conhecimentos em programação e, também, para programadores e estudantes que conhecem a linguagem C e querem aprofundar seus conhecimentos na linguagem de programação Python. Os exemplos práticos foram desenvolvidos utilizando a IDE (Integrated Development Environment) Eclipse com a adição da feature PyDev versão 8.2.0 (Python IDE for Eclipse) para o desenvolvimento dos códigos-fonte em Python e, para os códigos-fonte em linguagem C, da feature Eclipse Embedded C/C++ versão 6.1.2. A instalação destas features deve ser realizada na IDE Eclipse, no menu Help, na opção Eclipse Marketplace.

Aproveite!



Compartilhando conhecimento

2023

# SUMÁRIO

DOI: doi.org/10.36599/editpa-ipedfp

**Introdução à Programação ..... 7**

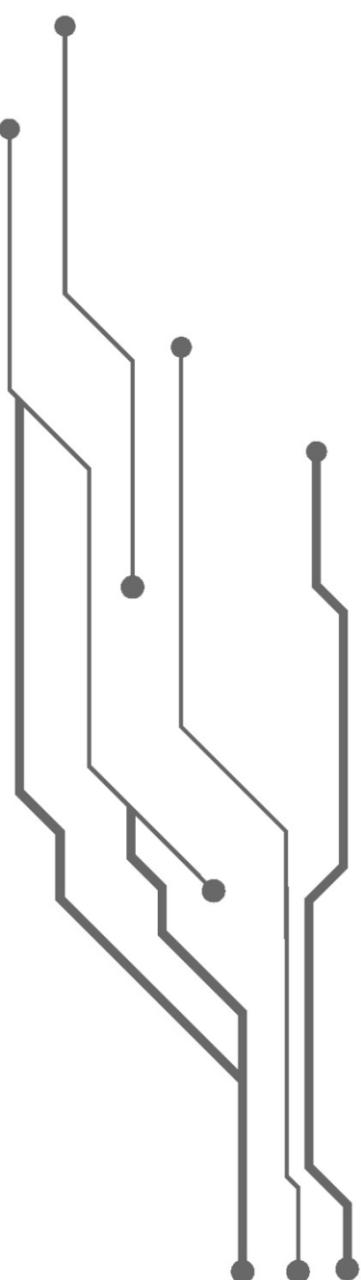
**Introdução à Programação em Python ..... 16**

**Principais comandos da Linguagem Python ..... 32**

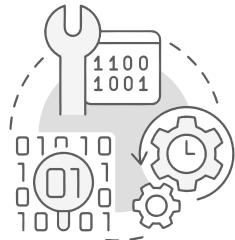
**Estruturas de Dados Fundamentais ..... 42**

**Funções ..... 48**

**Bibliografia ..... 57**



## CAPÍTULO 1



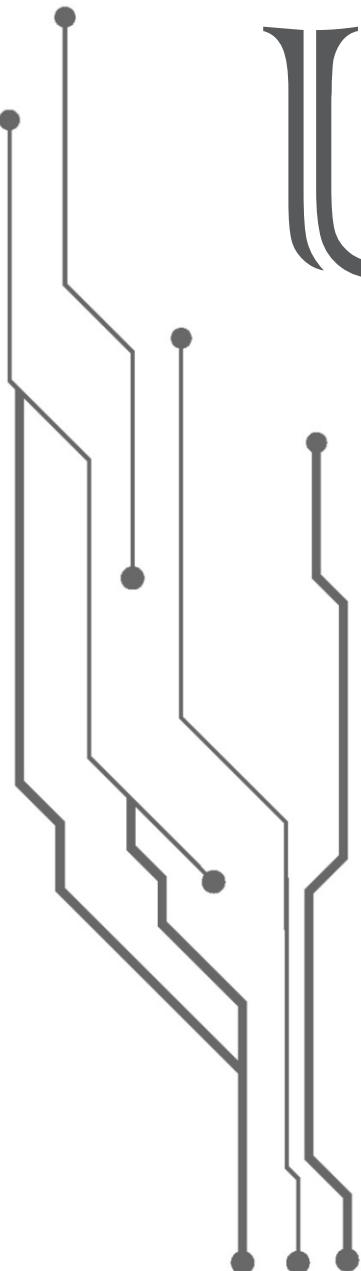
# Introdução à Programação

U[

m sistema computacional é dividido, basicamente, em duas partes: 1) hardware (parte física) e 2) software (parte lógica). O hardware abrange os componentes e o funcionamento interno do computador, enquanto que o software envolve o modo como se utiliza o computador. O hardware e o software são interdependentes, ou seja, de nada adianta um equipamento de última geração (hardware) sem que haja um bom programa (software) sendo executado nele, nem um software de última geração sendo executado em um computador ultrapassado (o que, na maioria das vezes, torna-se impossível) (SILVA et al., 2010; PARREIRA et al. 2017).

O processamento de dados realizado pelo computador (processamento de dados eletrônico) possui muitas vantagens sobre o processamento manual, principalmente pela sua velocidade de execução. Para que um computador possa desempenhar tarefas (processar dados) faz-se necessário (FALKEMBACH; SILVEIRA, 2005; PARREIRAS et al. 2017; SILVA et al., 2010):

1. Descrever antecipadamente as operações que serão efetuadas, prevendo-se todos os casos possíveis para o problema que deve ser resolvido. Esta especificação é realizada por meio de um **algoritmo**. Um algoritmo é um conjunto de regras que permite realizar mecanicamente todas as operações particulares correspondentes a uma determinada tarefa. O algoritmo é uma decomposição de um problema do



- usuário em operações elementares que podem ser executadas pelo computador utilizando alguma notação que programadores possam entender;
2. Traduzir o algoritmo obtido para uma linguagem de programação (programa fonte ou código-fonte). Uma linguagem de programação é composta por um conjunto bem definido de símbolos permitidos, regras de escrita e regras de comportamento (SEBESTA, 2018);
  3. Traduzir para o formato interno do computador essa sequência de operações, por meio de um programa especial chamado compilador. Um compilador é um programa que traduz instruções em linguagem de programação para o formato interno do computador (SEBESTA, 2018);
  4. Solicitar ao computador para que o programa seja executado, por meio de um comando do sistema operacional.

Uma segunda forma, mais completa, de apresentar os passos necessários para a resolução de problemas utilizando o computador é a seguinte (FALKEMBACH; SILVEIRA, 2005; PARREIRA et al. 2017; SILVA et al., 2010):

1. Definição do Problema: consiste na descrição da situação a ser resolvida, por meio de um enunciado que deve ser claro e completo, fornecendo todas as informações necessárias para a sua resolução;
2. Análise do problema: consiste em obter, a partir da definição, informações, ou seja, subsídios para construir um modelo para a solução do problema e formalizar este modelo por meio de mecanismos, tais como os algoritmos. Quanto mais complexo for o problema mais se recomenda a utilização desta técnica;
3. Programação ou Codificação: consiste em transcrever, em uma Linguagem de Programação, as instruções referentes ao modelo da solução, criando o código-fonte;
4. Edição: consiste em transferir para o computador, mais especificamente para a memória RAM (*Random Access Memory*), as instruções do código-fonte;
5. Compilação: consiste na interpretação das instruções do código-fonte por meio da verificação da sintaxe do programa. O processo de compilação gera o Programa Objeto (SILVEIRA et al., 2021);
6. Ligação (linkedição): a partir do Programa Objeto, gera-se o Programa Executável (SILVEIRA et al., 2021);
7. Análise dos resultados (fase de testes): consiste em verificar se os resultados correspondem à expectativa, pois o fato do computador

apresentar um resultado indica, inicialmente, que o programa não possui erros de sintaxe, mas não significa que não apresenta erros que envolvem a lógica de programação, ou seja, o programa pode estar escrito corretamente de acordo com as regras da linguagem de programação mas não produzir os resultados esperados;

8. Documentação: consiste em descrever os procedimentos que foram utilizados na resolução do problema, para utilizá-los na solução de problemas similares.

Uma linguagem de programação é um simbolismo que permite a comunicação entre o programador e o computador. A linguagem natural (como a Língua Portuguesa) não é útil para esta finalidade, pelo fato de que a sua sintaxe é muito complexa, assim como sua semântica (SILVEIRA et al., 2021).

Uma linguagem de programação é, portanto, mais restrita, e será definida por meio de um vocabulário autorizado, determinando sua sintaxe e sua semântica. Em uma linguagem de programação uma frase é chamada de instrução e corresponde à descrição de uma ou várias operações elementares do computador (SILVEIRA et al., 2021).

Atualmente, existem mais de 2000 linguagens de programação diferentes, a maioria delas utilizadas apenas em situações muito particulares. Para facilitar o estudo, estas linguagens são divididas em classes diferentes, de acordo com suas características principais. Estas diferentes classes são chamadas de paradigmas de programação (SEBESTA, 2018; SILVEIRA et al., 2021).

## 1.1 Conceitos Fundamentais

Para entendermos o que é um algoritmo, vamos partir de um exemplo. Queremos escrever um programa que calcule a média do rendimento acadêmico de um aluno. Supondo que, para ser aprovado, um aluno precisa atingir a média 7,0 (média aritmética simples) para ser aprovado (sem que seja necessário realizar o exame final), sendo que existem duas notas parciais, que iremos chamar de P<sub>1</sub> e P<sub>2</sub>. Sendo assim, a fórmula para o cálculo da média é a seguinte:

$$\text{Média} = \frac{P_1 + P_2}{2}$$

A média é igual à nota atribuída à 1<sup>a</sup> parte da avaliação (P<sub>1</sub>) mais a nota atribuída à 2<sup>a</sup> parte da avaliação (P<sub>2</sub>), sendo que este resultado é dividido por 2. Se a média for maior ou igual a 7,0 o aluno está aprovado; caso contrário, ainda poderá realizar a avaliação final (exame final).

A intenção é construirmos um algoritmo que automatize o cálculo da média. Um algoritmo é a série de passos que permite que o cálculo da média seja executado pelo computador. Basicamente, teríamos os seguintes passos:

1. obter as notas de P1 e P2;
2. aplicar a fórmula da média;
3. verificar a média calculada.

Estes passos estão descritos em linguagem natural. Cada um pode descrever os passos utilizando as palavras e a forma que achar melhor. Em programação isto não é possível pois, como vimos anteriormente, uma linguagem de programação é composta por um conjunto bem definido de símbolos permitidos e regras de escrita (sintaxe) e regras de comportamento (semântica), ou seja, para escrevermos um programa que automatize o cálculo da média, precisaremos seguir estas regras, caso contrário o programa não funcionará, ou seja, não será executado (SILVA et al., 2010; PARREIRA et al. 2017).

Uma das primeiras regras que precisamos entender é a utilização de informações variáveis e constantes. As informações variáveis precisam ficar armazenadas em posições da memória do computador ("gavetas") para que possam ser encontradas pelo programa. As variáveis são as informações que mudam no decorrer do tempo. No caso do programa que calcula a média, as informações variáveis são as notas de P1 e P2 e o resultado (média calculada). A Figura 1 apresenta, de forma gráfica, como as variáveis P1, P2 e média poderiam estar dispostas na memória do computador. Na figura foram representadas algumas gavetas vazias (podem existir inúmeras gavetas vazias, sendo limitadas à capacidade da memória utilizada) (SILVA et al., 2010; PARREIRA et al. 2017).



Figura 1 – Representação gráfica das variáveis na memória do computador  
(Adaptado de PARREIRA et al., 2017)

As informações constantes não variam durante a execução do programa. Por exemplo, no cálculo da média, independentemente da nota obtida pelo aluno, o resultado é sempre dividido por 2, pois estamos trabalhando com duas avaliações parciais para compor a média do aluno.

As informações variáveis, além de precisarem de um espaço de memória, precisam ter um tipo de dado associado, ou seja, o computador precisa saber

qual é o tipo de informação que cada uma das variáveis pode manipular. No caso da média de um aluno, o tipo de dado são os números reais, pois as notas são expressas com casas decimais. No início do algoritmo, o programador precisa declarar as variáveis que serão utilizadas no seu programa. O processo de declaração de variáveis consiste em definir um nome para as variáveis e o tipo de dado associado.

O nome de uma variável é um identificador (também conhecido como ID – *identifier*) e não muda durante a execução de um programa. Para definir o nome de uma variável devem ser seguidas algumas regras:

- não deve começar por um número;
- não deve conter espaços em branco;
- não deve conter caracteres especiais (+ - \* / % \$ # @ !), exceto o sublinhado (\_), que é utilizado para separar os nomes das variáveis;
- não deve conter nenhum caracter de acentuação;
- não pode ser uma palavra reservada da linguagem de programação (um comando da linguagem, por exemplo, é uma palavra reservada).

Seguem alguns exemplos de **nomes de variáveis válidos**:

- Endereco (sem o ç que é um caracter especial)
- Idade
- Nome
- Nome1
- Nome\_do\_Aluno
- P1
- Nota\_P1

Seguem alguns exemplos de **nomes de variáveis inválidos**:

- Nome do Aluno (contém espaços em branco)
- 1P (inicia por um número)
- Endereço (contém o caracter cedilha)
- if (é uma palavra reservada da linguagem, neste caso é uma palavra reservada das linguagens de programação C e Python que indica um comando ou instrução de seleção que estudaremos mais adiante).

Além do nome, na declaração de variáveis precisamos definir o tipo de dado que cada variável irá manipular. Os tipos de dados podem ser, por exemplo: números inteiros, números reais, cadeias de caracteres, entre outros. Estudaremos os tipos de dados das linguagens de programação C e Python mais adiante neste livro.

Antes de iniciarmos a codificação de um algoritmo precisamos, por meio da análise do problema, identificar as variáveis que serão necessárias (espaços de memória que serão necessários para armazenar os dados manipulados

pelo algoritmo). Estes dados podem ser divididos em dados de entrada e dados de saída.

Os dados de entrada são todas as informações que um programa precisa para chegar ao resultado. Estas informações são fornecidas por meio dos periféricos de entrada. O periférico de entrada mais comum é o teclado (o usuário digita as informações solicitadas pelo programa). Atualmente, com o uso de smartphones e o avanço dos dispositivos móveis, a tela do celular é muito utilizada como periférico de entrada, por meio da função touch screen ou tela de toque. Os dados de saída são os resultados obtidos por meio da execução de um programa (SILVA et al., 2010; PARREIRA et al. 2017). Quando falamos em dispositivos móveis, os programas são mais conhecidos como aplicativos ou pela sigla *app*.

No caso do algoritmo para calcular a média, os dados de entrada são as notas de P1 e P2 e o resultado é a média do aluno. Sendo assim, são necessários três espaços de memória para armazenar os dados. As três variáveis manipulam dados numéricos.

A declaração das variáveis para o algoritmo da média poderia ser a seguinte:

Var

P1, P2, Media: Real

A palavra reservada *Var* indica a seção de declarações de variáveis em um algoritmo, seguindo a sintaxe do VisuAlg, um ambiente visual para o estudo de algoritmos (PARREIRA et al., 2017) e estamos utilizando-o aqui neste texto apenas como exemplo. Cada ambiente e/ou linguagem de programação tem suas regras de sintaxe próprias. Sendo assim, a palavra *var* pode ou não ser utilizada em outros ambientes. Lembre-se que as variáveis não podem conter acentos.

Algumas observações importantes com relação às variáveis (SILVA et al., 2010; PARREIRA et al. 2017):

- 1) O uso do nome de uma variável, dentro de uma expressão (fórmula), não altera o seu valor. Para alterar o valor de uma variável na memória é preciso utilizar um comando da linguagem de programação que é o comando de atribuição (comando que indica que desejamos atribuir ou definir um valor para a variável, ou seja, que queremos modificar o valor que está armazenado na memória cujo espaço é representado pelo nome desta variável);
- 2) O uso do nome de uma variável, dentro de uma expressão (fórmula) significa que está sendo recuperado o seu conteúdo na memória;
- 3) Uma variável recebe o valor que lhe está sendo atribuído, perdendo o valor anteriormente armazenado (se guardamos,

anteriormente, o valor 7 na variável P1 e, por meio de um comando de atribuição, damos ordem para que seja guardado o valor 8, a variável P1 manterá apenas o último valor, o 8, ou seja, uma variável armazena apenas um valor de cada vez, não sendo possível resgatar um histórico dos valores armazenados. Caso precisemos armazenar mais de um valor utilizando um mesmo nome de variável, precisaremos utilizar uma estrutura de dados, tais como os vetores – ou matrizes unidimensionais, que serão estudados mais adiante neste livro).

Esta declaração de variáveis informa ao computador que o programa precisa de 3 espaços de memória (como se fossem 3 gavetas), que serão reservados durante a execução do programa. Este processo é denominado de alocação de memória. Reservar os espaços (gavetas) significa que outros programas e outras variáveis não poderão usar aqueles mesmos espaços, para que não haja o problema de conflito entre os dados, ou seja, dados de diferentes programas compartilhando as mesmas gavetas.

Para que possamos estudar como um algoritmo (programa) é executado pelo computador vamos estudar o funcionamento básico do computador. A Figura 2 apresenta um esquema básico de funcionamento do *hardware*, baseado na arquitetura de Von Neumann.

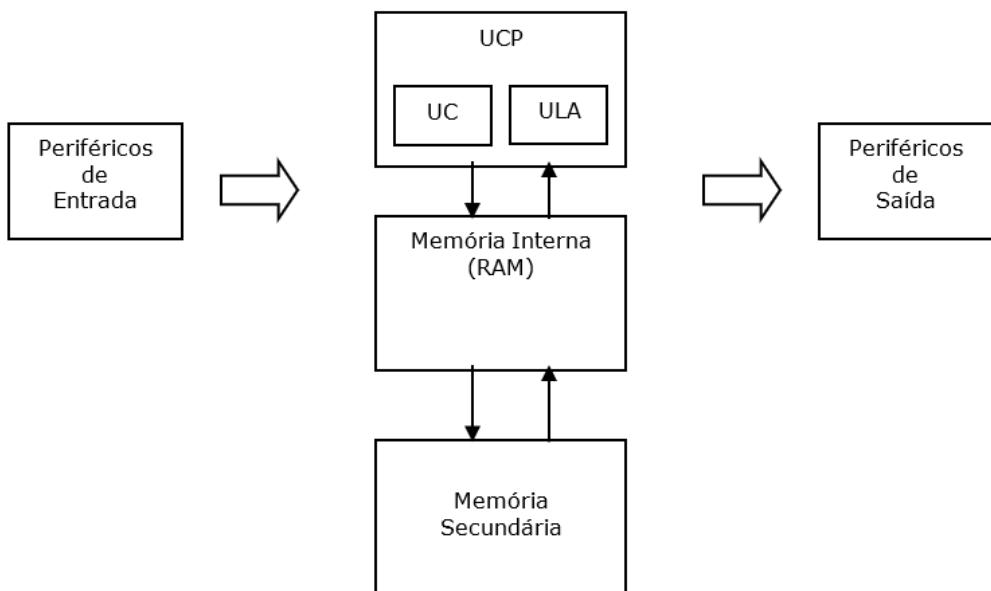


Figura 2 – Esquema Básico de Funcionamento do Computador de acordo com a Arquitetura de Von Neumann  
(Fonte: Os autores)

Conforme apresenta a Figura 2, temos as 3 fases do processamento de dados (entrada, processamento e saída) representadas, respectivamente: 1) pelos periféricos de entrada; 2) Unidade Central de Processamento e Memória e 3) periféricos de saída.

Os **periféricos de entrada** são todos os dispositivos que permitem que uma informação do meio externo seja enviada para o computador. Por exemplo, quando utilizamos um caixa eletrônico em um banco, podemos enviar informações de diferentes formas. Quando inserimos o cartão magnético, as informações são lidas por meio de uma leitora de cartões. Quando digitamos nossa senha, podemos usar o teclado ou o vídeo (touch screen).

O **processamento** é executado por meio de duas unidades principais, que são a UCP (Unidade Central de Processamento ou CPU – Central Process Unit) e a Memória. A UCP tem dois componentes principais, que são a UC (Unidade de Controle) e a ULA (Unidade Lógico-Aritmética). A UC é responsável por acompanhar o fluxo de execução do algoritmo/programa, indicando qual é o próximo comando a ser executado. A ULA, por sua vez, é responsável por interpretar as expressões aritméticas e lógicas. A memória é responsável por armazenar o algoritmo (ou programa), bem como as variáveis utilizadas. Quando o programa é colocado em execução, os valores das variáveis são enviados à UCP para serem utilizados nas expressões e são devolvidos à memória quando os resultados são calculados. Por isso é que existem setas indicando a mão dupla entre a Memória e a UCP. Com relação à Memória, a figura ilustra as memórias interna e secundária. A memória interna (ou Memória RAM Random Access Memory) armazena as instruções do programa que está sendo executado, bem como as variáveis que estão sendo utilizadas. Esta memória é volátil, ou seja, o programa, bem como as variáveis, só ficam armazenados enquanto o programa estiver sendo executado. O código-fonte do programa precisa ser armazenado (salvo) em uma memória secundária, para que possa ser executado outras vezes. Como exemplos de memórias secundárias temos o HD (Hard Disk ou Disco Rígido) e o pendrive.

O resultado do processamento é apresentado por meio de **periféricos de saída**. Os periféricos de saída mais comuns são o vídeo ou tela e a impressora. Por exemplo, após executar o algoritmo que calcula a média do aluno, a média é armazenada na memória. Se o programador não incluir no algoritmo/programa um comando para apresentar a média na tela, o usuário final (quem irá utilizar o seu programa) não visualizará o resultado.

Detalhando ainda mais a execução de um algoritmo, vimos que envolve 3 etapas (SILVA et al., 2010; PARREIRA et al. 2017):

- 1) Entrada de dados: nesta etapa, os dados são transmitidos de um meio externo para o computador. Normalmente, esta etapa é realizada por meio de um periférico de entrada. O periférico de entrada mais comum é o teclado mas, existem outros, tais como: scanner, leitora de códigos de barras, touch screen;
- 2) Processamento dos dados: esta etapa consiste na execução dos cálculos do programa, por meio da utilização dos dados armazenados na memória e da UCP. A memória interna e a UCP se comunicam, permitindo que os dados sejam enviados à ULA, processados e devolvidos para a memória, já com os resultados;

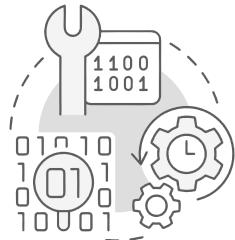
- 3) Saída de dados: esta etapa permite que os resultados obtidos sejam apresentados aos usuários, através de periféricos de saída. Os periféricos de saída mais comuns são o vídeo e a impressora.

No exemplo do algoritmo para calcular a média, estas 3 etapas ficariam assim distribuídas (SILVA *et al.*, 2010; PARREIRA *et al.* 2017):

- 1) Por meio de um periférico de entrada, como o teclado, o usuário informaria as notas de P1 e P2; estas notas seriam armazenadas na memória interna, nos espaços reservados;
- 2) Por meio da ULA, a média seria calculada de acordo com a fórmula adequada;
- 3) Por meio de um periférico de saída, como o vídeo, o usuário receberia o resultado, ou seja, a sua média.

Neste ponto cabe uma pergunta: Como é que o computador sabe que deve armazenar as notas de P1 e P2 na memória, calcular a média e depois mostrar o resultado? Estas ordens (comandos ou instruções) estão descritas no nosso algoritmo (programa). Este programa deve ser armazenado na memória do computador e, quando colocado em execução, aciona os dispositivos do *hardware*, necessários para o funcionamento do programa. Sendo assim, a memória, além de armazenar os dados (de entrada e de saída) também armazena o programa (a série de passos que deve ser seguida para que se chegue à solução do problema).

## CAPÍTULO 2



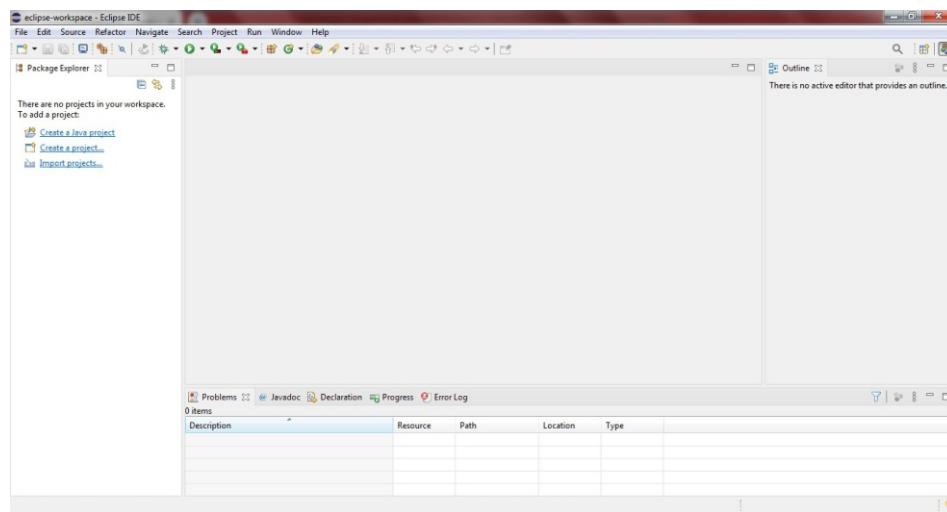
# Introdução à Programação em Python

N

este capítulo iremos estudar alguns conceitos introdutórios sobre a linguagem de programação Python, apresentando os principais comandos da linguagem, comparando-os aos comandos similares da linguagem de programação C.

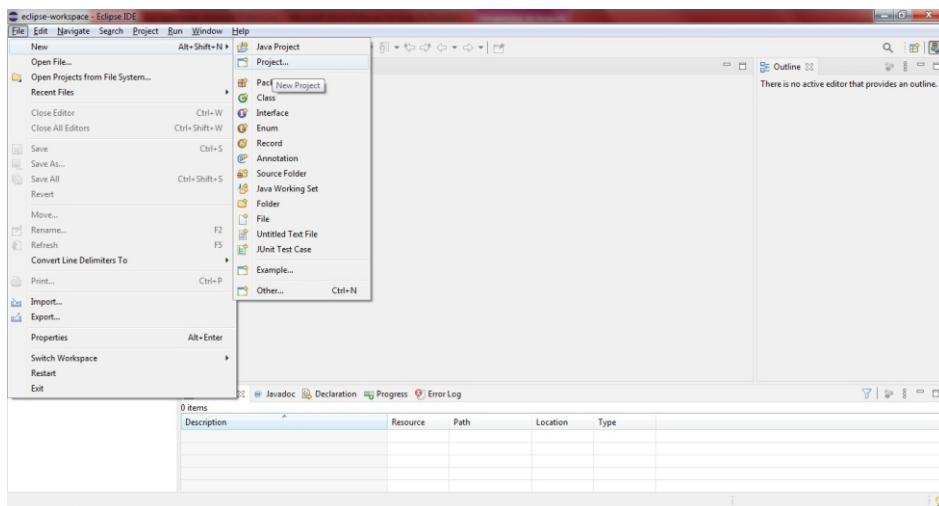
Vamos iniciar o estudo introdutório da linguagem de programação Python criando um primeiro programa, utilizando a IDE Eclipse, apresentada na Figura 3. Tradicionalmente, quando iniciamos o estudo de uma nova linguagem de programação (já conhecendo os conceitos de algoritmos e lógica de programação), criamos um programa denominado *Hello World* ou Olá Mundo. Para criarmos este programa vamos utilizar o comando print, para mostrarmos uma mensagem na tela.

Figura 3: Interface da IDE Eclipse (Fonte: Os autores)



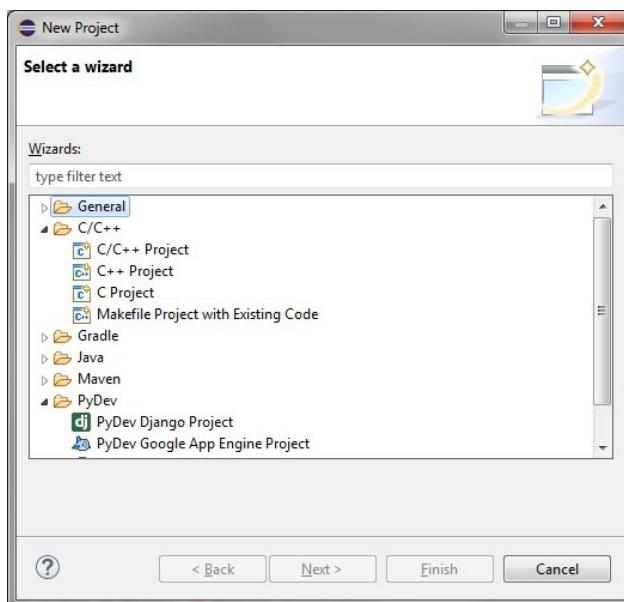
No Eclipse vamos selecionar a opção New -> Project no menu File, como mostra a Figura 4.

Figura 4: Menu File opção New -> Project (Fonte: Os autores)



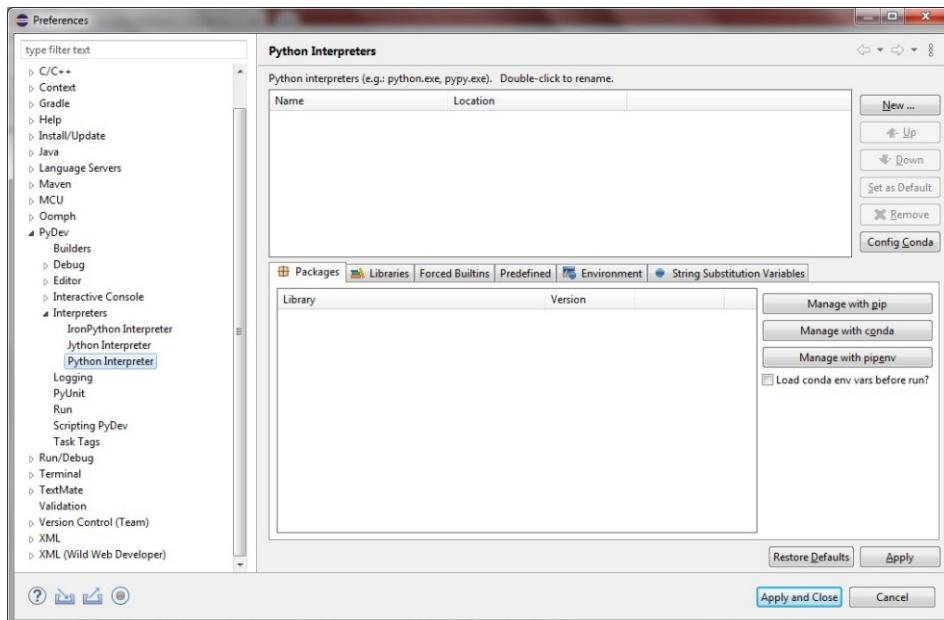
Como instalamos na nossa IDE as features para o desenvolvimento de programas nas linguagens C e Python, veremos as opções de diferentes projetos, tais como CProject e PyDev Project (como mostra a Figura 5).

Figura 5: Escolha do Tipo de Projeto na IDE Eclipse (Fonte: Os autores)



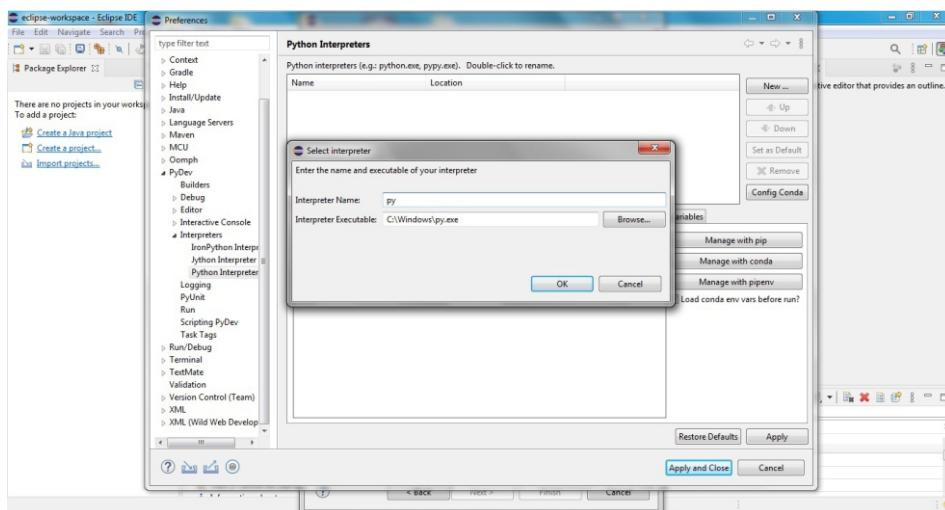
Vamos fazer o primeiro exemplo utilizando a linguagem de programação Python, a partir da criação de um novo projeto do tipo PyDev Project. Antes de criar o projeto é preciso configurar o interpretador de Python que será utilizado, por meio da opção *Python Interpreters* do menu *Windows -> Preferences* no Eclipse. Essa opção é apresentada na Figura 6.

Figura 6: Configuração do Interpretador Python (Fonte: Os autores)



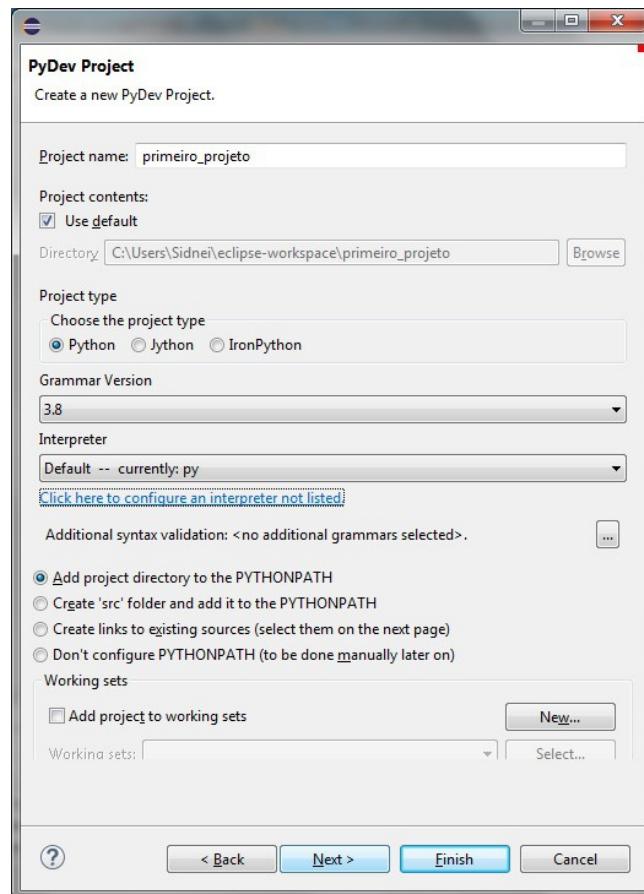
Ao selecionar esta opção na janela apresentada na Figura 6 devemos clicar no botão New e selecionar a pasta onde se encontra o arquivo executável do interpretador Python que está instalado no nosso computador, como mostra a Figura 7.

Figura 7: Localização (Pasta) do Interpretador Python (Fonte: Os autores)



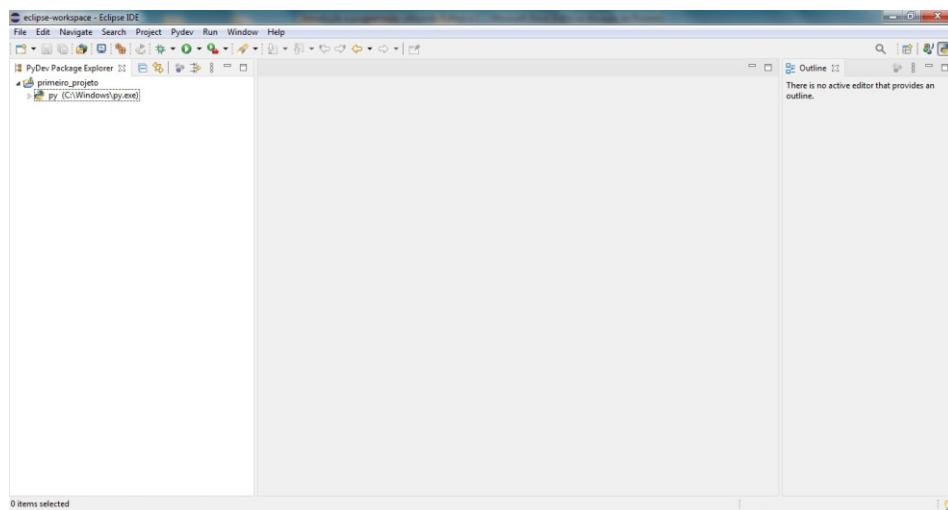
Após selecionarmos o tipo de projeto (conforme Figura 5) devemos clicar no botão Next e digitarmos um novo para o nosso projeto e clicarmos no botão Finish, como mostra a Figura 8 (escolhemos o nome *primeiro\_projeto*).

Figura 8: Definição do Nome do Projeto (Fonte: Os autores)



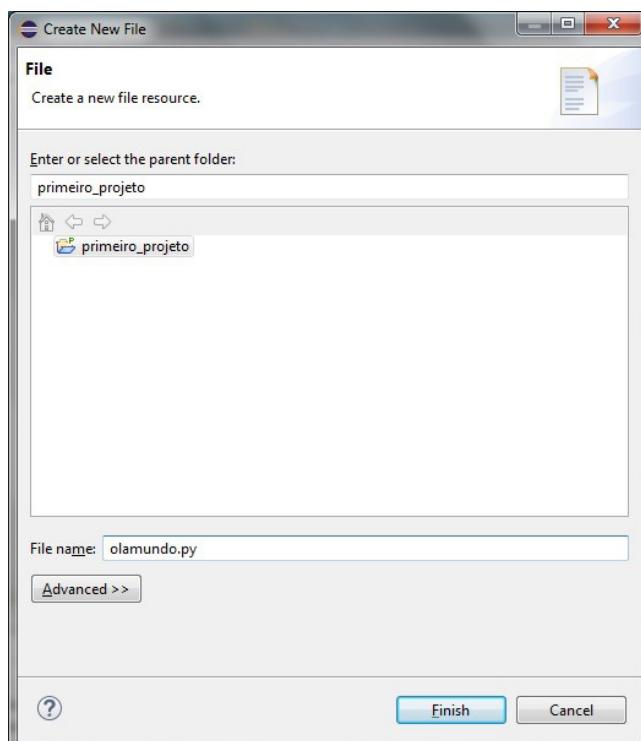
Após a criação do projeto que denominamos de *primeiro\_projeto* veremos a tela apresentada na Figura 9.

Figura 9: Tela Inicial do Primeiro Projeto (Fonte: Os autores)



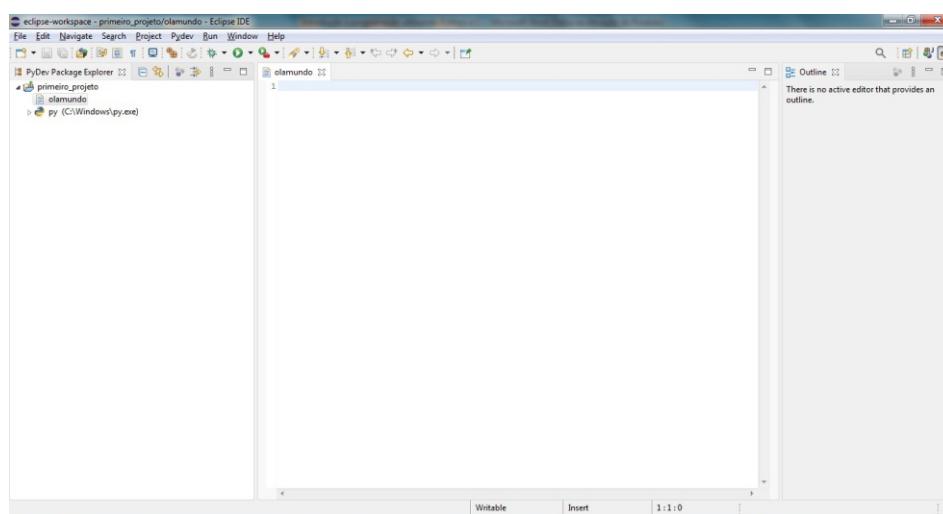
Agora, por meio da opção *New -> File* do menu *File*, vamos inserir um arquivo que conterá o código-fonte em Python do nosso primeiro programa. A Figura 10 mostra a janela que será aberta ao selecionarmos esta opção. Nesta janela devemos selecionar o nome do nosso projeto (*primeiro\_projeto*) e definirmos o nome do nosso arquivo (utilizamos o nome *olamundo.py* como exemplo, sendo que *py* é a extensão dos arquivos contendo o código-fonte em Python).

Figura 10: Criação de um novo arquivo (Fonte: Os autores)



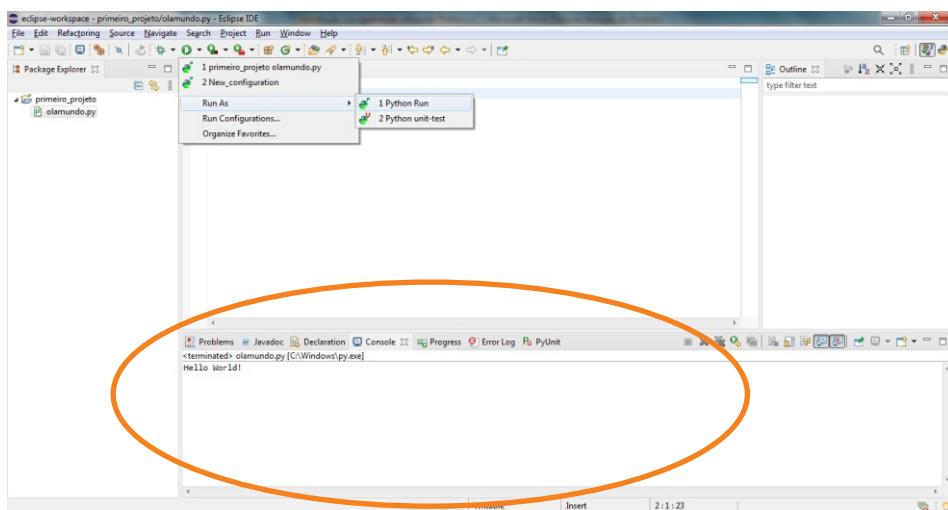
Agora temos um espaço na tela onde iremos digitar o código-fonte do nosso primeiro programa em Python (Figura 11).

Figura 11: Novo arquivo criado no projeto (Fonte: Os autores)



Vamos digitar o comando `print("Hello World!")` e executar o programa, usando a opção Run As -> Python Run do menu Run, como mostra a Figura 12. O resultado do comando `print` será apresentado na janela de console, na parte inferior da tela (destacada na Figura 12).

Figura 12: Execução do primeiro programa (Fonte: Os autores)



Comparando o código-fonte para mostrar a mensagem *Hello World* utilizando as linguagens de programação Python e C, como mostra o quadro 1, vemos que a linguagem Python é mais simples. Na Linguagem C precisamos incluir a biblioteca Standard IO (Input-Output) para utilizarmos o comando `printf` por meio da instrução `#include <stdio.h>` e, além disso, o programa precisa ter uma rotina principal (`main`), além do bloco de comandos delimitado pelas chaves.

Quadro 1 – Comparação dos Códigos-Fonte

Python	C
<code>print("Hello World!")</code>	<code>#include &lt;stdio.h&gt;</code> <code>main () {</code> <code>printf("Hello World!");</code> <code>}</code>

Fonte: Os autores

## 2.1 Características Importantes da Linguagem de Programação Python

Algumas características importantes, para iniciarmos o estudo da linguagem de programação Python, compreendem:

- não é necessário declarar as variáveis. Basta atribuir um valor para uma variável que a mesma será automaticamente declarada (declaração implícita) e assumirá o tipo de dado de acordo com o que for atribuído à mesma;

Por exemplo:

Valor = 10

A variável *Valor* (ou identificador *Valor*) recebe o valor inteiro 10 e a variável passa a ter o tipo de dados inteiro atrelado à mesma

Nome = "José"

A variável *Nome* recebe a *string* (cadeia de caracteres) José

- a linguagem Python é *case sensitive*, ou seja, há diferenciação em utilizar letras maiúsculas ou minúsculas. Por exemplo: os identificadores *Valor* e *valor* não são iguais, pois *Valor* inicia com letra maiúscula e *valor* com letra minúscula. Neste caso são dois identificadores diferentes;
- a indentação faz parte da sintaxe da linguagem. Por exemplo, na instrução abaixo, há um erro de sintaxe, pois o comando *print* não está corretamente indentado, ou seja, não faz parte do comando *if*

If x>8:

    print("x é maior do que 8")

O código-fonte correto é (o comando *print* tem que estar dentro do *if*):

If x>8:

    print("x é maior do que 8")

Após digitar o comando *if* e pressionar a tecla *Enter* o Python já irá indentar a próxima linha.

- nas expressões condicionais que compõem as instruções *if*, *for* e *while*, o uso de parênteses é opcional.

O Quadro 2 apresenta a comparação das principais características entre Python e C.

Quadro 2 – Comparação das Principais Características

Python	C
Não é necessário declarar as variáveis	A declaração de variáveis é obrigatória
A linguagem é <i>case sensitive</i> (diferencia letras maiúsculas e minúsculas)	A linguagem é <i>case sensitive</i>
A indentação faz parte da sintaxe	A indentação não faz parte da sintaxe (são utilizados os símbolos {} para os blocos de comandos)
O uso do ponto -e-vírgula (;) no final de uma instrução é opcional. Entretanto, se existir mais de uma instrução em uma mesma linha do código-fonte, o ponto -e-vírgula passa a ser obrigatório. Por exemplo:  <code>print("Soma",2+2);print("Multiplicação",2*2)</code>	O uso do ponto -e-vírgula (;) no final de uma instrução é obrigatório

Fonte: Os autores

A Figura 13 mostra alguns exemplos de atribuição de valores a variáveis e impressão dos valores das mesmas, bem como de alguns resultados de expressões.

Figura 13: Exemplos em Python (Fonte: Os autores)

The screenshot shows the Eclipse IDE interface. In the top-left corner, there's a small icon of a circle with a dot. The main window has a title bar 'eclipse-workspace - teste1/codigo1.py - Eclipse IDE'. Below the title bar is a menu bar with 'File', 'Edit', 'Refactoring', 'Source', 'Source Refactor', 'Navigate', 'Project', 'Run', 'Window', and 'Help'. On the left side, there's a 'Project Explorer' view showing a project named 'teste1' with files 'codigo1.py' and 'py (C:\Windows\py.exe)'. The central area contains a code editor with the following Python code:

```

1 valor=10
2 Valor=15
3
4 print(valor)
5 print(Valor)
6 print(valor+Valor)
7
8 nome="José"
9 print(nome)
10
11 sobrenome="Silva"
12 print(nome+sobrenome)
13

```

To the right of the code editor is a 'Console' view showing the output of the code execution:

```

10
15
25
José
JoséSilva

```

Analisando os exemplos da Figura 13, temos:

- atribuição do número inteiro 10 à variável *valor* (variável criada com letras minúsculas);
- atribuição do número inteiro 15 à variável *Valor* (note que são variáveis diferentes, pois o 2º nome *valor* está com a letra V em maiúsculo);

- impressão dos conteúdos das variáveis *valor* e *Valor* (resultados apresentados na janela de console, na parte inferior da tela);
- impressão da soma dos conteúdos das variáveis *valor* e *Valor*;
- atribuição do nome *José* à variável *nome*;
- atribuição do sobrenome *Silva* à variável *sobrenome*;
- impressão da soma (concatenação) das strings *nome* e *sobrenome*.

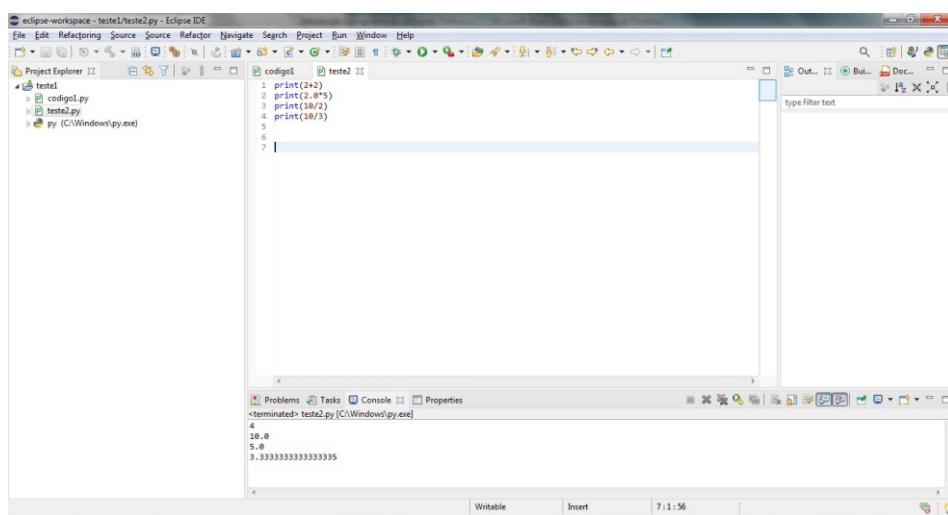
## 2.2 Operadores Aritméticos e Expressões Aritméticas

Os operadores aritméticos básicos da linguagem Python são: + (adição), - (subtração), \* (multiplicação) e / (divisão). Estes operadores são os mesmos da linguagem de programação C. Podemos utilizar o comando *print* para mostrar o resultado de operações aritméticas. Por exemplo, vamos testar os comandos abaixo na IDE Eclipse:

```
print(2+2)
print(2.0*5)
print(10/2)
print(10/3)
```

A Figura 14 apresenta os comandos e o resultado da execução (na janela de console) na IDE Eclipse.

Figura 14: Exemplos de Operações Aritméticas (Fonte: Os autores)



Podemos colocar uma mensagem junto com as operações aritméticas, por exemplo:

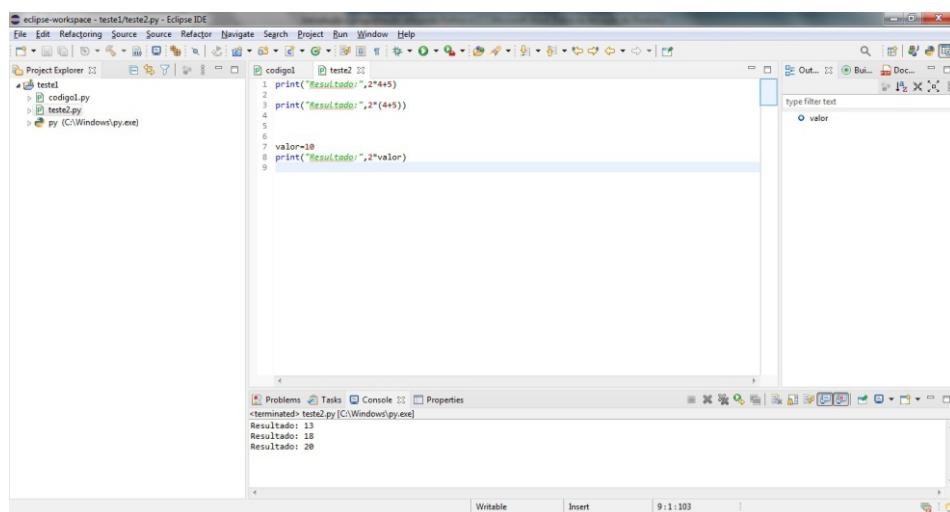
```
print("Resultado",2*4+5)
print("Resultado",2*(4+5))
```

Também podemos utilizar variáveis na impressão dos resultados das expressões aritméticas. Por exemplo:

```
valor=10
print("Resultado:",2*valor)
```

A Figura 15 apresenta os resultados dos exemplos executados na IDE Eclipse.

Figura 15: Exemplos de Operações Aritméticas (Fonte: Os autores)



Quadro 3 – Comparação das Principais Características

<b>Python</b>	<b>C</b>
<code>print(2+2)</code>	<code>#include &lt;stdio.h&gt; main () {     printf("%i",2+2); }</code>
<code>print(2.0*5)</code>	<code>#include &lt;stdio.h&gt; main () {     printf("%f",2.0*5); }</code>
<code>print(10/2)</code>	<code>#include &lt;stdio.h&gt; main () {     printf("%i",10/2); }</code>

Continua

Continuação

<b>Python</b>	<b>C</b>
<code>print(10/3)</code>	<code>#include &lt;stdio.h&gt;</code>  <code>main () {</code> <code>printf("%d",10/3);</code> <code>}</code>
<code>print("Resultado",2*4+5)</code>	<code>#include &lt;stdio.h&gt;</code>  <code>main () {</code> <code>printf("Resultado:%d",2*4+5);</code> <code>}</code>
<code>print("Resultado",2*(4+5))</code>	<code>#include &lt;stdio.h&gt;</code>  <code>main () {</code> <code>printf("Resultado:%d",2*(4+5));</code> <code>}</code>

Fonte: Os autores

Analisando os exemplos de código-fonte em linguagem C, apresentados no Quadro 2, vemos que é preciso formatar a saída de dados do comando `printf` utilizando o símbolo `%` seguido do tipo de dados que será impresso, tal como `d` para um valor decimal inteiro, ou `f` para um valor fracionário (`float` ou `double`). Este mesmo tipo de formatação também pode ser utilizado na linguagem Python, como veremos em um exemplo mais adiante nesta seção.

Podemos utilizar os argumentos `sep` e `end` para formatar a saída de um comando `print`. O argumento `sep` coloca um espaço em branco (valor padrão) entre os argumentos que serão impressos, ou outro símbolo que podemos definir. O valor padrão do argumento `end` é uma nova linha “`\n`”. Sendo assim, o comando `print` adiciona uma nova linha depois da impressão mas podemos definir outro valor, tal como uma tabulação por exemplo, utilizando “`\t`”.

A Figura 16 apresenta o resultado da execução de um exemplo alterando o valor de `sep` para dois traços (--) e o valor de `end` para três pontos (...).

Figura 16: Utilização dos argumentos `sep` e `end` com o comando `print` (Fonte: Os autores)

```

eclipse-workspace - teste1/codigo1.py - Eclipse IDE
File Edit Refactoring Source Source Refactor Search Project Run Window Help
Project Explorer Problems Tasks Console Properties
> teste1
  > codig1
1 # alterando os valores de sep e end na função print
2
3 valores = '1000'
4 valores = '2000'
5 valores = '3000'
6 valores = '4000'
7
8 texto = "Alterando o valor de sep para dois traços"
9 print(texto)
10 print(valores, valores, valores, valores, sep='--')
11
12 # pode usar \n
13 print()
14
15 texto = "Alterando o valor de sep para dois traços e de end para três pontos"
16 print(texto)
17 print(valores, valores, valores, valores, sep='--', end='...\\n')
18
19 |

```

Console output:

```

Alterando o valor de sep para dois traços
1000--2000--3000--4000
Alterando o valor de sep para dois traços e de end para três pontos
1000--2000--3000--4000...

```

Vamos fazer um exemplo utilizando o símbolo % para formatar a saída do comando *print*. Vamos fazer o cálculo da raiz quadrada, utilizando uma função matemática disponível na biblioteca *Math*. A biblioteca *Math* permite a utilização de diversas funções matemáticas, tais como os cálculos de raiz quadrada, seno e cosseno, por exemplo. Para utilizar esta biblioteca é preciso importá-la, por meio do comando *import math*.

Por exemplo, supondo que queiramos calcular a raiz quadrada de um número, podemos utilizar o código-fonte a seguir.

```
import math

numero=input('Digite um número:')

numero=int(numero)

print('A raiz quadrada de ',numero,' é igual a ',math.sqrt(numero))
```

Supondo que o usuário digite o valor 10 na entrada de dados (*input*), será mostrado o seguinte resultado:

*A raiz quadrada de 10 é igual a 3.1622776601683795*

Podemos formatar a saída do comando *print*, definindo o número de casas (algarismos) antes e depois da vírgula (ponto decimal). Por exemplo:

```
print('A raiz quadrada de ',numero,' é igual a %3.2f.'%math.sqrt(numero))
```

onde *%3.2f*. significam 3 algarismos antes do ponto decimal, 2 após o ponto decimal e *f* significa um número do tipo *float*. A impressão do resultado ficará assim:

*A raiz quadrada de 10 é igual a 3.16.*

## 2.3 Operadores Relacionais e Lógicos

Os operadores relacionais da linguagem Python são: *>* (maior que), *>=* (maior ou igual), *<* (menor que), *<=* (menor ou igual), *==* (igual), e *!=* (diferente), sendo os mesmos da linguagem de programação C (SILVA et al., 2010). Vejamos alguns exemplos de aplicação na Figura 17. Na janela de console temos apresentados os resultados da avaliação das respectivas expressões lógicas (*True* ou *False*).

De acordo com o código-fonte, temos a variável *a* com o valor 5 e a variável *b* com o valor 10. Para cada expressão lógica utilizamos o comando *print* para mostrar o resultado da sua avaliação. Por exemplo: *a>10* resulta *False* pois a expressão lógica é falsa, já que o valor da variável *a* é menor do que 10.

Figura 17: Exemplos de Aplicação dos Operadores Relacionais em Python (Fonte: Os autores)

The screenshot shows the Eclipse IDE interface with a Python project named 'teste1'. In the 'teste2' editor, the following code is displayed:

```

1 a=5
2 b=10
3 print(a<b)
4 print(b>a)
5 print(b==b)
6 print(b!=a)
7 print(a>b)
8 print(a==b)
9 print(a==b)
10 print(a!=b)
11
12

```

In the 'Console' tab, the output of the script 'teste2.py' is shown:

```

<terminated> teste2.py [C:\Windows\py.exe]
False
False
True
True
False
False
True
True

```

Devemos lembrar que o símbolo de igualdade (`=`) não significa uma operação de comparação e, sim, de atribuição de um valor a uma variável. Por exemplo:

`a=10` # o valor 10 está sendo atribuído à variável `a`

`a==10` # estamos perguntando (comparando), se o valor da variável `a` é igual a 10

Os operadores lógicos mais comuns são o `and` e o `or` e seu funcionamento é igual ao de outras linguagens de programação. Quando o `and` (e) é utilizado, todas as condições devem ser satisfeitas para que a expressão lógica retorne verdadeiro (`True`). Utilizando o operador lógico `or` (ou), pelo menos uma das condições deve ser satisfeita para que a expressão lógica retorne `True`. Veja os exemplos da Figura 18, em que utilizamos o comando `print` para mostrar o resultado da avaliação das expressões lógicas.

Figura 18: Exemplos de Aplicação dos Operadores Relacionais e Lógicos em Python (Fonte: Os autores)

The screenshot shows the Eclipse IDE interface with a Python project named 'teste1'. In the 'teste2' editor, the following code is displayed:

```

1 a=10
2 b=5
3 c=7
4 print(a>b and b<c)
5 print(a>b or b<c)
6 print(a>c and b>c)
7 print((a>c) and (b>c))
8 print(((a>c) or (b>a)) and b>c)
9 print(((a>c) or (b>a)) and b>c)
10 print(((a>c) or (b>a)) and b>c)
11
12

```

In the 'Console' tab, the output of the script 'teste2.py' is shown:

```

<terminated> teste2.py [C:\Windows\py.exe]
True
True
False
True
True
False
False

```

Analizando os exemplos da Figura 18 e os resultados da avaliação das expressões lógicas, apresentadas na janela de console, temos (sendo que  $a$  é igual a 10,  $b$  é igual a 5 e  $c$  tem armazenado o valor 7):

- $a>b \text{ and } b<a$ :  $a$  é maior que  $b$  – True e  $b < a$  – True, True and True resulta True, de acordo com a tabela verdade (SILVA et al., 2010);
- $a>b \text{ or } b<a$ :  $a$  é maior que  $b$  – True e  $b < a$  True, True or True resulta True;
- $a>c \text{ and } b>c$ :  $a$  é maior que  $c$  – True e  $b$  é maior que  $c$  – False, True and False retorna False;
- $a < c \text{ or } b < c$ :  $a$  é menor que  $C$  – True e  $b$  é menor do que  $c$  – True, True or True retorna True;
- $(a>c \text{ and } b<a) \text{ or } b>c$ : neste exemplo usamos os parênteses para delimitar (ou definir a ordem de avaliação, pois as expressões entre parênteses serão avaliadas primeiro) as expressões, ou seja, a resposta da primeira expressão com and será comparada com a segunda expressão com or:  $a$  é maior que  $C$  – True e  $b$  é menor do que  $a$ , também resulta True – True and True retorna True. A expressão  $b$  é maior que  $c$  retorna False. Finalmente, True or False retorna True;
- $(a < c \text{ or } b < a) \text{ or } b > c$ : neste exemplo usamos os parênteses para delimitar as expressões, ou seja, a resposta da primeira expressão com or será comparada com a segunda expressão, também usando or:  $a$  é menor que  $C$  (False) e  $b$  é menor do que  $a$  (True) – False or True retorna True. A expressão  $b$  é maior que  $c$  retorna False. Finalmente, True or False retorna True;
- $(a < c \text{ or } b < a) \text{ and } b > c$ : neste exemplo usamos os parênteses para delimitar as expressões, ou seja, a resposta da primeira expressão com or será comparada com a segunda expressão, usando and:  $a$  é menor que  $C$  (False) e  $b$  é menor do que  $a$  (True) – False and True retorna False. A expressão  $b$  é maior que  $c$  retorna False. Finalmente, True and False retorna False.

O Quadro 4 apresenta a comparação dos exemplos de expressões lógicas entre Python e C.

Quadro 4 – Comparação das Expressões Lógicas Exemplificadas

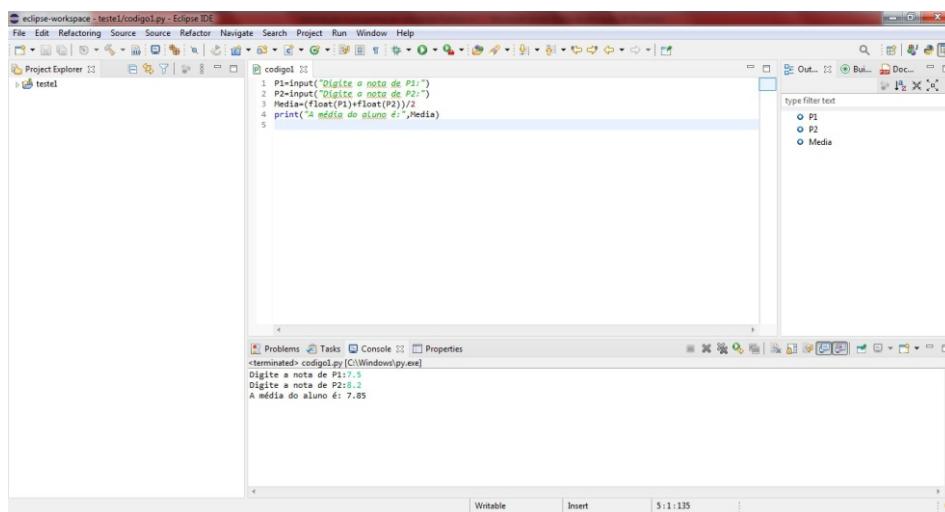
<i>Python</i>	<i>C</i>
$a>b \text{ and } b<a$	$(a>b \&\& b < a)$
$a>b \text{ or } b<a$	$(a>b \mid\mid b < a)$
$a>c \text{ and } b>c$	$(a>c \&\& b > c)$
$a < c \text{ or } b > c$	$(a < c \mid\mid b > c)$
$(a>c \text{ and } b<a) \text{ or } b>c$	$((a>c \&\& b < a) \mid\mid b > c)$
$(a < c \text{ or } b < a) \text{ or } b > c$	$((a < c \mid\mid b < a) \mid\mid b > c)$
$(a < c \text{ or } b < a) \text{ and } b > c$	$((a < c \mid\mid b < a) \&\& b > c)$

Fonte: Os autores

## 2.4 Entrada e Saída de Dados

Já vimos nos exemplos deste capítulo que o comando `print` é um dos comandos utilizados em Python para a saída de dados, pois temos apresentado os resultados dos exemplos na janela de console da IDE Eclipse. Vamos relembrar o exemplo do algoritmo para calcular a média de um aluno, utilizado no capítulo 1. Para calcular a média, precisamos das duas notas do aluno (P1 e P2) e, para tal, precisamos utilizar um comando para a entrada de dados. Vamos utilizar, então, o comando `input` que permite a digitação dos valores na janela de console. Os valores digitados por meio do comando `input` são do tipo `string` em Python. Sendo assim, para utilizarmos as notas de P1 e P2 no cálculo da média, precisaremos convertê-las para o tipo numérico desejado. Como as notas de um aluno podem conter casas decimais, vamos utilizar o tipo `float`. Na Figura 19 apresentamos o código-fonte e a execução com um exemplo de cálculo da média para um aluno que obteve as notas 7.5 para P1 e 8.2 para P2.

Figura 19: Exemplo de entrada de dados com o comando `input` (Fonte: Os autores)



O código-fonte, apresentado de forma detalhada, é o seguinte:

`P1=input("Digite a nota de P1:") # entrada de dados – permite que o usuário informe, na janela de console da IDE Eclipse, a nota obtida na 1a avaliação e a armazena na variável P1 (o símbolo de igualdade significa a atribuição à variável)`

`P2=input("Digite a nota de P2:") # segunda entrada de dados – armazena a nota digitada na variável P2`

`Media=(float(P1)+float(P2))/2 # realiza o cálculo da média aritmética. Como P1 e P2, por terem sido informadas pelo comando input, são do tipo string, não podem ser utilizados em uma expressão aritmética com divisão. Sendo assim, utilizando a função float, convertemos os valores de P1 e P2 para um valor numérico com casas decimais. Se os`

valores fossem apenas inteiros poderíamos utilizar a função `int`. Como a divisão por 2 só deve ocorrer após a soma das duas notas, colocamos a soma entre parênteses, para definir a ordem de execução do cálculo.

`print("A média do aluno é:",Media) # saída de dados, onde apresentamos a mensagem “A média do aluno é:” e a média que foi calculada. Note que usamos a primeira letra maiúsculas nas variáveis. Lembre-se que a linguagem Python é case sensitive.`

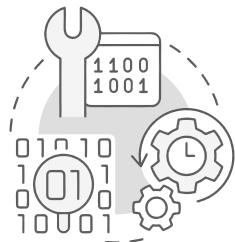
O Quadro 5 apresenta a comparação do exemplo para calcular a média codificada em Python e em C.

Quadro 5 – Comparação do Código-Fonte para Calcular a Média

Python	C
<pre>P1=input("Digite a nota de P1:") P2=input("Digite a nota de P2:") Media=(float(P1)+float(P2))/2 print("A média do aluno é:",Media)</pre>	<pre>#include &lt;stdio.h&gt; main() {     float P1, P2, Media;     scanf("%f", &amp;P1);     scanf("%f",&amp;P2);     Media=(P1+P2)/2;     printf("A média do aluno é: %2.f \n", Media); }</pre>

Fonte: Os autores

## CAPÍTULO 3



# Principais comandos da Linguagem Python

## 3.1 Expressões Condicionais e Comandos de Seleção

JP

Por meio de expressões condicionais (ou condições), os desenvolvedores de software podem definir condições para que um determinado comando (ou blocos de comandos) sejam ou não executados. Por exemplo, se pensarmos no programa para calcular a média de um aluno podemos definir uma condição para mostrar se o aluno está ou não aprovado e, em caso de não estar aprovado, definir uma ação que deva ser tomada (fazer uma prova de recuperação ou uma exame final). Um dos comandos mais comuns, em diferentes linguagens de programação, utilizado para avaliar o resultado de expressões condicionais é o comando *se* (*if*) e existem diferentes formas de utilização, como veremos nesta seção.

### 3.1.1 Seleção Simples

Uma seleção simples ocorre quando verificamos apenas se a expressão condicional (ou condição) é verdadeira, sendo executado o comando (ou bloco de comandos) logo abaixo da condição. Por exemplo, no comando abaixo, se o resultado da avaliação da expressão condicional *condicao1* for verdadeiro, será executado o comando definido após os dois pontos. Caso contrário, se o resultado for falso, o fluxo de execução do programa continuará na próxima instrução, na linha seguinte à comando. A sintaxe do comando *if* em Python é apresentada a seguir. Lemos o comando da seguinte forma: *se a condicao1 for verdadeira, então comando será executado*.

```
if condicao1:  
    comando
```

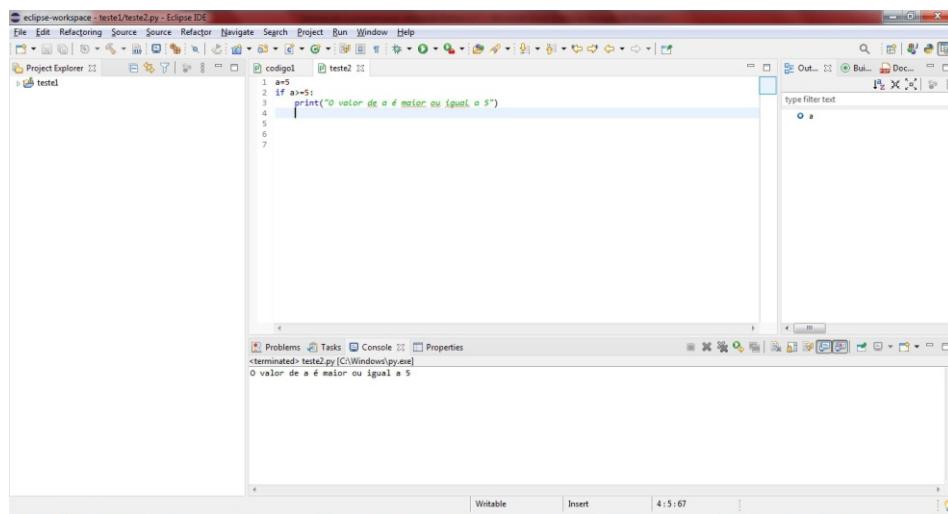


Se o resultado da condição `if a >= 5` for True o comando da linha abaixo, desde que esteja corretamente indentado, será executado. Lembre-se que, em Python, a indentação faz parte das regras de sintaxe da linguagem. Por exemplo:

```
a=5
if a>=5:
    print("o valor de a é maior ou igual a 5")
```

A Figura 20 mostra o exemplo de seleção simples executado na IDE Eclipse. Como o resultado da expressão é True, já que a variável “a” tem o valor igual a 5, a mensagem do comando `print` é exibida na janela de console.

Figura 20: Exemplo de seleção simples (Fonte: Os autores)



O Quadro 6 apresenta a comparação do exemplo codificado em Python e em C.

Quadro 6 – Comparação do Código-Fonte para Calcular a Média

<i>Python</i>	<i>C</i>
<pre>a=5 if a&gt;=5:     print("O valor de a é maior ou igual a 5")</pre>	<pre>#include &lt;stdio.h&gt; int a; main () {     a=5;     if (a&gt;=5) {         printf("O valor de a é maior ou igual a 5");     } }</pre>

Fonte: Os autores

### 3.1.2 Seleção Composta

Uma seleção composta ocorre quando verificamos se a expressão condicional (ou condição) é verdadeira ou falsa. Sendo assim, será executado o comando (ou bloco de comandos) logo abaixo da condição, caso a mesma seja verdadeira (`comando1` no exemplo abaixo) ou após o `else`, se a condição for falsa. Por exemplo, no comando abaixo, se o resultado da avaliação da expressão condicional `condicao1` for verdadeiro, será executado o comando definido após os dois pontos. Caso contrário, se o resultado for falso, o fluxo de execução do programa continuará na próxima instrução, na linha seguinte à comando. A sintaxe do comando `if` em Python é apresentada a seguir. Podemos ler essa instrução da seguinte forma: se a `condicao1` for verdadeira então o `comando1` será executado, senão, caso a `condicao1` seja falsa, o `comando2` será executado. Seguindo as regras de sintaxe da linguagem Python, devemos colocar dois pontos (`:`) no final da expressão condicional e depois do comando `else` também.

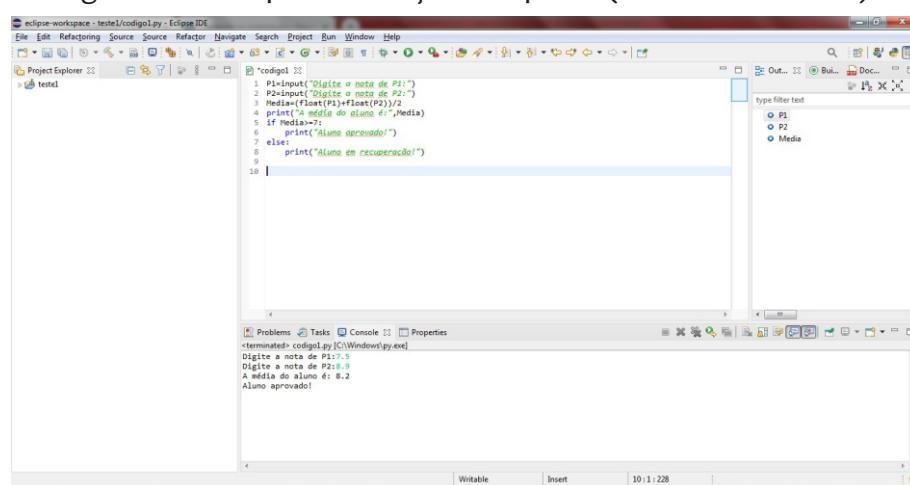
```
if condicao1:  
    comando1  
else:  
    comando2
```

Por exemplo, se fizermos uma expressão condicional para avaliarmos se um aluno foi aprovado, utilizando a seleção composta, teríamos:

```
if media>=7:  
    print("Aluno aprovado")  
else:  
    print("Aluno em recuperação")
```

Inserimos este comando de seleção composta no exemplo que fizemos no capítulo anterior (Figura 19). A Figura 21 apresenta o resultado da execução deste exemplo modificado.

Figura 21: Exemplo de seleção composta (Fonte: Os autores)



O Quadro 7 apresenta a comparação do exemplo da Figura 21 codificado em Python e em C.

Quadro 7 – Comparação do Código-Fonte para Calcular a Média

Python	C
<pre>P1=input("Digite a nota de P1:") P2=input("Digite a nota de P2:") Media=(float(P1)+float(P2))/2 print("A média do aluno é:",Media) if Media&gt;=7:     print("Aluno aprovado!") else:     print("Aluno em recuperação!")</pre>	<pre>#include &lt;stdio.h&gt; main() {     float P1, P2, Media;     scanf("%f", &amp;P1);     scanf("%f", &amp;P2);     Media=(P1+P2)/2;     printf("A média do aluno é: %2.f \n", Media);     if (Media&gt;=7) {         printf("Aluno aprovado!");     }     else {         printf("Aluno em recuperação!");     } }</pre>

Fonte: Os autores

### 3.1.3 Seleção de Múltipla Escolha

Uma seleção de múltipla escolha ocorre quando queremos verificar várias condições verdadeiras em um mesmo comando de seleção, ou seja, é utilizada para a verificação de vários casos. Em Python utilizamos, para uma seleção de múltipla escolha, os comandos *if* e *elif*, com a seguinte sintaxe:

```
if condicao1:
    comando1
elif condicao2:
    comando2
elif condicao3:
    comando3
else:
    comando4
```

Se a *condicao1* for verdadeira, será executado o *comando1*. Se a *condicao2* for verdadeira, será executado o *comando2* e assim por diante, testando as condições verdadeiras para todos os comandos *elif*. Caso nenhuma das condições seja verdadeira será executado o comando após *else*.

Por exemplo, vamos criar um programa que apresenta o percentual da alíquota de Imposto de Renda de acordo com o salário de uma pessoa, seguindo a tabela hipotética em que: até R\$2.500,00 o contribuinte está isento de Imposto de Renda, acima de R\$2.500,00 até R\$3.500,00 a alíquota é de 10%, entre R\$3.501,00 e R\$4.500,00 a alíquota é de 15%, entre R\$4.501,00 e R\$5.500,00 a alíquota é de 20% e, acima de R\$5.500,00 a alíquota é de 25%. Utilizando a seleção de múltipla escolha temos o seguinte código-fonte, apresentado na Figura 22 (note que não estamos fazendo o cálculo do valor do imposto a pagar e, sim, apenas mostrando qual o percentual de alíquota devido). Lembre-se que os valores informados por meio do comando `input` são do tipo `string`. Sendo assim, para testarmos o valor da variável `Salario` nas condições, fazemos a conversão da mesma para o tipo `float`.

Figura 22: Exemplo de seleção de múltipla escolha (Fonte: Os autores)

```

eclipse-workspace - teste1/codigo1.py - Eclipse IDE
File Edit Refactoring Source Source Refactor Navigate Search Project Run Window Help
Project Explorer Build Target Document
> teste1
codigo1.py
1 Salario=input("Digite o valor do salário:")
2 if float(Salario)<=2500:
3     print("Isento de Imposto de Renda!")
4 elif float(Salario)>2500 and float(Salario)<=3500:
5     print("A alíquota de Imposto de Renda é de 10%")
6 elif float(Salario)>3500 and float(Salario)<=4500:
7     print("A alíquota de Imposto de Renda é de 15%")
8 elif float(Salario)>4500 and float(Salario)<=5500:
9     print("A alíquota de Imposto de Renda é de 20%")
10 else:
11     print("A alíquota de Imposto de Renda é de 25%")
12
13
14

```

Problems Tasks Console Properties

<terminated> codigo1.py[C:\Windows\temp]

Digite o valor do salário:12000

A alíquota de Imposto de Renda é de 25%

Na linguagem de programação C o comando `switch` pode ser utilizado para realizar uma seleção de múltipla escolha. Entretanto, este comando permite a utilização de valores do tipo inteiro (`int`) ou caractere (`char`), e o salário que utilizamos no exemplo é um valor do tipo `float`. Sendo assim, vamos fazer um outro exemplo para comparar a seleção de múltipla escolha em Python e C. Vamos supor que, por meio de uma nota, entre 1 e 5, podemos representar o conceito de um curso. O Quadro 8 apresenta o código-fonte deste exemplo de seleção de múltipla escolha, comparando as linguagens de programação Python e C. Como a variável `nota`, na linguagem Python, é informada por meio do comando `input`, o seu tipo de dados é `string`. Para utilizarmos na comparação dos conceitos, poderíamos manter a nota como `string` ou convertermos para um número inteiro (utilizando `int`) como fizemos no exemplo apresentado no Quadro 8.

Quadro 8 – Comparação do Exemplo de Seleção de Múltipla Escolha

<i>Python</i>	<i>C</i>
<pre>nota=input("Digite a nota do curso:") if int(nota)==1:     print("Conceito Insuficiente") elif int(nota)==2:     print("Conceito mínimo") elif int(nota)==3:     print("Conceito médio") elif int(nota)==4:     print("Conceito Bom") elif int(nota)==5:     print("Conceito Muito Bom") else:     print("Nota não classificada nos conceitos")</pre>	<pre>#include &lt;stdio.h&gt; int nota; main () { printf("Digite a nota do curso:"); scanf("%i",&amp;nota); switch(nota) { case 1: printf("Conceito insuficiente"); break; case 2: printf("Conceito mínimo"); break; case 3: printf("Conceito médio"); break; case(4): printf("Conceito Bom"); break; case(5): printf("Conceito Muito Bom"); break; default: printf("Nota não classificada nos conceitos"); } }</pre>

Fonte: Os autores

## 3.2 Comandos de Repetição

### 3.2.1 Comando For

O comando *for* permite percorrer os itens de uma coleção (por exemplo, uma *string* ou uma estrutura de dados do tipo *lista*). Para cada um dos itens da coleção, será executado o comando declarado na estrutura de repetição (*loop*). Ao percorrer a lista de valores, a variável definida no comando *for* receberá, a cada iteração, um dos itens da coleção. A sintaxe do comando *for* em Python é:

```
for variavel in lista:
    comandos
```

Por exemplo, utilizando o seguinte trecho de código:

```
nomes=['Ana','João','José','Maria']
for n in nomes:
    print(n)
```

Inicialmente criamos uma lista, denominada `nomes`, contendo quatro nomes (*Ana, João, José e Maria*). Após, utilizando o comando `for`, vamos percorrer toda a lista e, a cada iteração, um dos nomes, sequencialmente, será armazenado na variável `n`, que será impressa na tela. O resultado da execução deste trecho de código será a impressão dos nomes conforme estão armazenados na lista: *Ana, João, José e Maria*.

Podemos incluir o comando `else` ao final do comando `for`. Assim, um comando ou bloco de comandos será executado ao final da iteração. Por exemplo:

```
nomes=['Ana','João','José','Maria']
for n in nomes:
    print(n)
else:
    print("A lista de nomes foi impressa com sucesso!")
```

Também podemos usar o comando `for` para escrever todas as letras de um *string* (como se fosse soletrar as letras). Isso é possível porque uma *string* é uma lista de caracteres. Por exemplo:

```
for caracteres in 'Python':
    print('Letra:', caracteres)
```

Este trecho de código exibirá, como resultado:

*Letra: P*

*Letra: y*

*Letra: t*

*Letra: h*

*Letra: o*

*Letra: n*

Outra forma de utilização do comando `for` é por meio de intervalos de repetição, definidos por `range`. Por exemplo:

```
for i in range(3):
    print(i)
```

Este laço de repetição será executado três vezes, no intervalo de 0 a 2, sendo `i` igual a 0 na primeira interação.

Vamos fazer um exemplo a partir do programa para calcular a média de um aluno, implementado anteriormente. Vamos utilizar o comando `for` para calcular a média de um grupo de 10 alunos e, ao final, mostrar o número de alunos aprovados (média  $\geq 7$ ) e o número de alunos em recuperação.

A Figura 23 apresenta esse exemplo na IDE Eclipse.

Figura 23: Exemplo com o comando `for` (Fonte: Os autores)

```
eclipse-workspace - teste1/codigo1.py - Eclipse IDE
File Edit Refactoring Source Source Refactor Navigate Search Project Run Window Help
Project Explorer Build Target Document Properties Problems Tasks Console Properties
.codigo1
1 aprovados=0
2 for contador in range(10):
3     P1=input("Digite a nota de P1:")
4     P2=input("Digite a nota de P2:")
5     Media=(float(P1)+float(P2))/2
6     print("A média do aluno é:",Media)
7     if Media>=7:
8         aprovados+=1
9     print("Número de alunos aprovados:",aprovados)
10 print("Número de alunos em recuperação:", 10-aprovados)
11
12

Problems Tasks Console Properties
<terminated> codigo1.py [C:\Windows\py.exe]
A média do aluno é: 8.1
Digite a nota de P1:8
Digite a nota de P2:9
A média do aluno é: 8.5
Digite a nota de P1:7
Digite a nota de P2:7.2
A média do aluno é: 7.1
Digite a nota de P1:6
Digite a nota de P2:6.5
A média do aluno é: 6.25
Número de alunos aprovados: 7
Número de alunos em recuperação: 3
```

O Quadro 9 apresenta a comparação do exemplo da Figura 23 em Python com a linguagem de programação C.

Quadro 9 – Comparação do Exemplo com o Comando For

<b>Python</b>	<b>C</b>
<pre> aprovados=0 for contador in range(10):     P1=input("Digite a nota de P1:")     P2=input("Digite a nota de P2:")     Media=(float(P1)+float(P2))/2     print("A média do aluno é:",Media)     if Media&gt;=7:         aprovados=aprovados+1 print("Número de alunos aprovados") print("Número de alunos em recuperação:", 10 - aprovados) </pre>	<pre> #include &lt;stdio.h&gt; main() {     float P1, P2, Media;     int contador, aprovados;     for (contador=1; contador&lt;11; contador++) {         printf("Digite a nota de P1:");         scanf("%f", &amp;P1);         printf("Digite a nota de P2:");         scanf("%f", &amp;P2);         Media=(P1+P2)/2;         printf("A média do aluno é: %.2f \n", Media);         if (Media&gt;=7) {             aprovados=aprovados+1;         }     }     printf("Número de alunos aprovados %d:",aprovados);     printf("Número de alunos em recuperação %d:",10-aprovados); } </pre>

Fonte: Os autores

### 3.2.2 Comando While

O comando *while* faz com que um trecho de código seja repetido enquanto uma condição for verdadeira. Quando o resultado da condição for falso, a execução da repetição é interrompida (saindo do *loop*) e passando para o próximo comando após o *while*. A sintaxe do comando *while* em Python é:

```

while (condição):
    comandos

```

Por exemplo:

```

contador=0
while(contador<5):
    print(contador)
    contador=contador+1

```

Neste exemplo de código estamos repetindo, por meio do `while`, o comando `print` e incrementando o contador 5 vezes, utilizando uma variável contadora (`contador`). Inicializamos a variável `contador` com o valor zero. Enquanto o valor do contador for menor do que 5 (enquanto o resultado desta condição for `True`), o trecho dentro do `while` será repetido. Lembre-se que a indentação do código-fonte faz parte das regras de sintaxe da linguagem Python. Sendo assim, os comandos dentro do `while` devem estar corretamente indentados para que sejam executados na repetição. O resultado da execução deste trecho é apresentado a seguir:

```
0  
1  
2  
3  
4
```

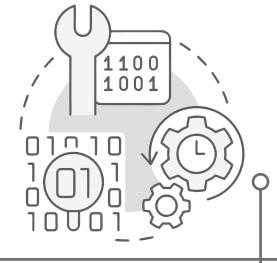
O Quadro 10 apresenta a comparação deste exemplo em Python com a linguagem de programação C.

Quadro 10 – Comparaçāo do Exemplo com o Comando While

<i>Python</i>	<i>C</i>
<pre>contador=0 while (contador&lt;5):     print(contador)     contador=contador+1</pre>	<pre>#include &lt;stdio.h&gt; main() {     int contador;     contador=0;     while (contador&lt;5) {         printf(contador);         contador=contador+1;     } }</pre>

Fonte: Os autores

## CAPÍTULO 4



# Estruturas de Dados Fundamentais

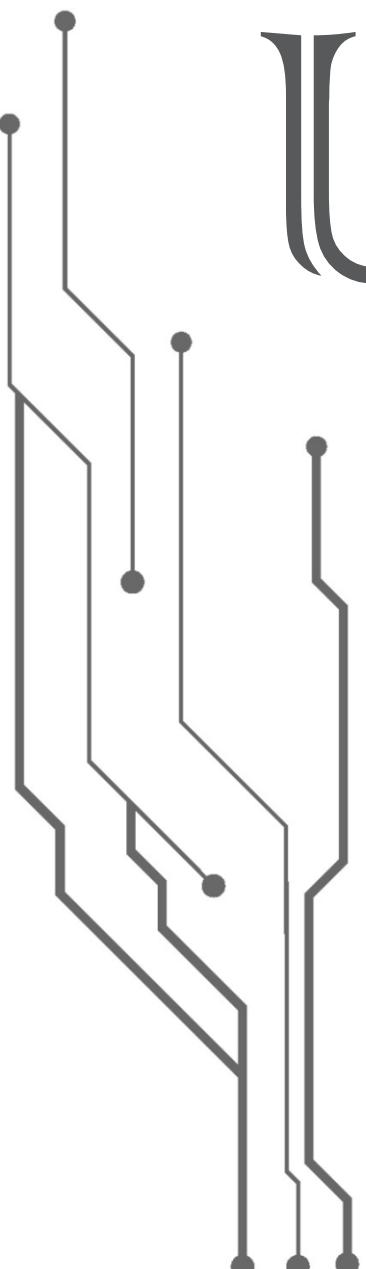
U

Uma variável, em um programa pode ser simples ou composta. Nos três primeiros capítulos deste e-book utilizamos apenas variáveis simples, que possuem um identificador e um único valor. As variáveis compostas podem ser homogêneas ou heterogêneas, possuem um identificador e armazenam mais de um valor. As variáveis compostas são **estruturas de dados**. Uma estrutura de dados envolve, além da estrutura de armazenamento, as regras de definição, manipulação e de acesso aos dados contidos nas mesmas. Os programas, por meio de estruturas de dados, organizam logicamente os dados, permitindo sua visualização de diferentes formas (SILVA et al., 2010).

As variáveis compostas homogêneas referem-se a uma estrutura de dados (ou conjunto de dados) que armazenam dados do mesmo tipo (inteiro, caractere, etc.) e esses dados são acessados por meio de um índice (valor numérico inteiro) que representa a posição do dado dentro da estrutura. As principais estruturas de dados homogêneas são os vetores (também conhecidos como matrizes unidimensionais ou matrizes de uma dimensão) e as matrizes bidimensionais.

### 4.1 Vetores (ou Listas em Python)

Um vetor (ou matriz unidimensional), na linguagem Python, é uma lista. As listas são definidas entre colchetes, sendo seus elementos separados por vírgula. Os elementos podem ser acessados por meio do índice que indica a posição do elemento desejado na lista, iniciando pelo índice zero. O índice -1 corresponde ao último item da lista. Sucessivamente, o índice -2 permite acessar o penúltimo item da lista e assim por diante.



Vamos fazer alguns exemplos criando e manipulando listas utilizando a IDE Eclipse. Vamos criar um novo programa em Python com os seguintes exemplos de código-fonte:

```

vetor=[1,2,3,4,5,6,7,8,9,10]
nomes=["Ana","João","Maria","Pedro"]
salarios=[1100,2500,3200,5000]

print("Valor da primeira posição do vetor:",vetor[0])
print("Valor da Última posição do vetor:",vetor[-1])
print("Terceiro nome da lista nomes:",nomes[2])
print("Primeiro salário da lista salarios:",salarios[0])
print("Lista de nomes armazenados:")
for i in nomes:
    print("Nome:",i)

```

A Figura 24 apresenta este código-fonte, bem como seus respectivos resultados de execução, utilizando a IDE Eclipse.

Figura 24: Exemplos de Listas/Vetores (Fonte: Os autores)

The screenshot shows the Eclipse IDE interface with the following details:

- Code Editor:** The main window displays the Python code from Figure 24.
- Console Tab:** The bottom tab bar has tabs for "Problems", "Tasks", "Console", and "Properties". The "Console" tab is active, showing the output of the program's execution:

```

Valor da primeira posição do vetor: 1
Valor da Última posição do vetor: 10
Terceiro nome da lista nomes: Maria
Primeiro salário da lista salarios: 1100
Lista de nomes armazenados:
Nome: Ana
Nome: João
Nome: Maria
Nome: Pedro

```

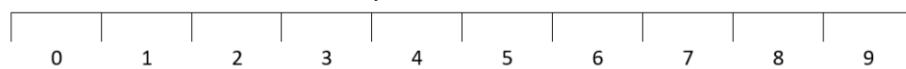
Vamos utilizar o exemplo para calcular a média de 10 alunos (conforme Quadro 9, do capítulo anterior). Vamos utilizar vetores (listas em Python) para armazenar, separadamente, as notas da 1<sup>a</sup> avaliação dos 10 alunos, as notas da 2<sup>a</sup> avaliação e as respectivas médias. Assim teremos três vetores, como mostra a Figura 25.

Figura 25 – Representação Gráfica dos Vetores (Fonte: Os autores)

Vetor P1 – Notas da 1<sup>a</sup> avaliação (lembrando que os índices, em Python, iniciam em zero)



Vetor P2 – Notas da 2<sup>a</sup> avaliação



**Vector Media – Médias aritméticas das notas dos 10 alunos**



A Figura 26 apresenta o exemplo de código-fonte em Python, utilizando os vetores apresentados anteriormente. Como vamos armazenar as médias no vetor *Media*, após o laço *for* da entrada de dados podemos acessar as médias individualmente (bem como as notas de P1 e de P2), o que não era possível no exemplo que fizemos no capítulo 3, já que estávamos utilizando variáveis simples.

Figura 26: Exemplo de Cálculo da Média com 3 vetores/listas  
(Fonte: Os autores)

The screenshot shows the Eclipse IDE interface with the following details:

- File Path:** eclipse-workspace / testel/codigo1.py - Eclipse IDE
- Toolbar:** File Edit Refactoring Source Source Refactor Navigate Search Project Run Window Help
- Project Explorer:** Shows a folder named 'testel'.
- Code Editor:** Displays the content of 'codigo1.py':

```
1 #!/usr/bin/python
2 alunos = [9.0, 8.0, 9.0, 8.0, 9.0, 8.0]
3 Media = sum(alunos)/len(alunos)
4 print("Média dos alunos:",Media)
5
6 for contador in range(10):
7     aluno = float(input("Digite a nota de P1 do aluno:{}\n".format(contador+1)))
8     P1[contador] = aluno
9     aluno = float(input("Digite a nota de P2 do aluno:{}\n".format(contador+1)))
10    P2[contador] = aluno
11    Media[contador] = (P1[contador]+P2[contador])/2
12    print("A média do aluno:{} é:{}".format(contador+1,Media[contador]))
13    if Media[contador] >= 6.0:
14        aprovados += 1
15    print("Número de alunos aprovados:",aprovados)
16    print("Número de alunos em recuperação:",10-aprovados)
17
18 print("Médias dos alunos:")
19 for contador in range(10):
20     print("Nota do aluno:{} é:{}".format(contador+1,Media[contador]))
```

- Output Console:** Shows the execution results:

```
[Eclipse Java EE IDE for Web Developers - 2018-12 (4.8.0)] Problems Tasks Console Properties
/home/luiz/Downloads/codigo1.py [C:\Downloads\py.exe]
Digite a nota de P1 do aluno:9
Digite a nota de P2 do aluno:9
Média do aluno: 9.0
Número de alunos aprovados: 9
Número de alunos em recuperação: 1
Médias dos alunos:
Média do aluno: 1.0
Média do aluno: 2.395
Média do aluno: 3.0
Média do aluno: 4.0
Média do aluno: 5.975
Média do aluno: 6.0
Média do aluno: 7.0
Média do aluno: 8.845
Média do aluno: 9.0
Média do aluno: 10.0
```

O Quadro 11 apresenta a comparação do exemplo do cálculo da média com vetores em Python e em Linguagem C.

Quadro 11 – Comparação do Exemplo com Vetores/Listas

<i>Python</i>	<i>C</i>
<pre> aprovados=0 P1=[0,0,0,0,0,0,0,0,0] P2=[0,0,0,0,0,0,0,0,0] Media=[0,0,0,0,0,0,0,0,0]  for contador in range(10):     print("Aluno:",contador+1)     P1[contador]=float(input("Digite a nota de P1 do aluno:"))     P2[contador]=float(input("Digite a nota de P2 do aluno:"))     Media[contador]=(P1[contador]+P2[contador])/2 print("A média do aluno é:",Media[contador]) if Media[contador]&gt;=7:     aprovados=aprovados+1  print("Número de alunos aprovados:",aprovados) print("Número de alunos em recuperação:",       10-aprovados) print("Médias dos alunos:") for contador in range(10):     print("Média do aluno:",           contador+1,Media[contador]) </pre>	<pre> #include &lt;stdio.h&gt; main() {     float P1[10], P2[10], Media[10];     int contador, aprovados;     for (contador=0; contador&lt;10; contador++) {         printf("Digite a nota de P1.");         scanf("%f", &amp;P1[contador]);         printf("Digite a nota de P2.");         scanf("%f", &amp;P2[contador]);         Media[contador]=(P1[contador]+P2[contador])/2;         printf("A média do aluno é: %.2f\n",                Media[contador]);         if (Media[contador]&gt;=7) {             aprovados=aprovados+1;         }     }     printf("Número de alunos aprovados %d:",aprovados);     printf("Número de alunos em recuperação %d:",10-aprovados);     printf("Média dos alunos:");     for (contador=0; contador&lt;10; contador++) {         printf("Média do aluno: %d %.2f\n",                contador,Media[contador]);     } } </pre>

Fonte: Os autores

## 4.2 Matrizes Bidimensionais (Listas em Python)

Uma matriz bidimensional, na linguagem Python, é uma lista com dois índices, geralmente denominados de *linha* e *coluna*, como uma tabela ou uma planilha do Microsoft Excel. As matrizes são definidas entre colchetes, sendo os elementos de cada linha inseridos entre colchetes (separados por vírgulas) e, cada linha também é separada por vírgulas.

Por exemplo, a seguinte atribuição em Python cria uma matriz A com 2 linhas e 2 colunas:

$$A = [[10, 20], [30, 40]]$$

A Figura 27 mostra a representação gráfica da matriz bidimensional A ( $2 \times 2$  – 2 linhas e 2 colunas), lembrando que os índices, em Python, iniciam no valor zero).

Figura 27: Representação Gráfica da Matriz A (Fonte: Os autores)

	Coluna 0	Coluna 1
Linha 0	10	20
Linha 1	30	40

Para percorrermos a matriz A precisamos, então, de dois índices (*linha* e *coluna*). O código-fonte abaixo mostra como percorrer e imprimir os valores contidos na matriz A em Python:

```
A=[[10,20],[30,40]]  
for linha in range(2):  
    for coluna in range(2):  
        print("Valores armazenados na matriz A:",A[linha][coluna])
```

Vamos fazer o exemplo do cálculo da média utilizando uma matriz bidimensional para armazenar as notas de P1, de P2 e as médias. Ao invés de utilizarmos três vetores (matrizes unidimensionais) como fizemos na seção 4.1, vamos criar uma matriz bidimensional com 3 linhas (P1, P2 e Média) e 10 colunas (uma coluna para cada aluno). A linha 0 será utilizada para armazenar as notas de P1, a linha 1 para as notas de P2 e a linha 2 para armazenar as médias de cada aluno. A Figura 28 mostra a representação gráfica desta matriz  $3 \times 10$  (3 linhas e 10 colunas).

Figura 28: Representação Gráfica da Matriz Bidimensional 3x10  
(Fonte: Os autores)

A horizontal number line starting at 0 and ending at 9. There are 10 major tick marks labeled 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Between each labeled tick mark, there are 9 unlabeled grid lines, creating a total of 10 segments or intervals of length 1 unit each.

A Figura 29 apresenta o exemplo construído na IDE Eclipse. Criamos uma matriz bidimensional com 3 linhas e 10 colunas denominada de *Notas*. Armazenamos, na linha 0, os valores de P1 (*Notas[0][coluna]*), os valores de P2 na linha 1 e o cálculo da média na linha 2.

Figura 29: Exemplo de Matriz Bidimensional para Calcular a Média dos Alunos  
 (Fonte: Os autores)

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace / testel / codigo1.py - Eclipse IDE
- Menu Bar:** File Edit Refactoring Source Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard icons for file operations like Open, Save, Print, etc.
- Project Explorer:** Shows a single project named "testel".
- Code Editor:** Displays the content of "codigo1.py":

```
Notas=[[0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0]]
for columna in range(10):
    print("Aluno ",columna+1)
    Notas[0][columna]=float(input("Digite a nota de P1 do aluno:"))
    Notas[1][columna]=float(input("Digite a nota de P2 do aluno:"))
    Notas[2][columna]=(Notas[0][columna]+Notas[1][columna])/2
    if Notas[2][columna]>=7:
        print("Aluno aprovado")
    else:
        print("Aluno em recuperacao")
--
```

- Terminal:** Shows the execution of the script and its output:

```
Média do aluno 3 8.75
Aluno aprovado
Média do aluno 4 4.75
Aluno em recuperacao
Média do aluno 5 2.5
Aluno em recuperacao
Média do aluno 6 9.5
Aluno aprovado
Média do aluno 7 8.1
Aluno aprovado
Média do aluno 8 9.8
Aluno aprovado
Média do aluno 9 9.8
Aluno aprovado
Média do aluno 10 5.8
Aluno em recuperacao
```
- Status Bar:** Writable Insert 20 : 1 : 574

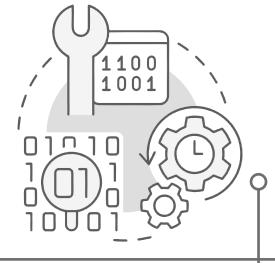
O Quadro 12 apresenta a comparação do exemplo do cálculo da média com uma matriz bidimensional com 3 linhas e 10 colunas em Python e em Linguagem C.

Quadro 12 – Comparação do Exemplo com Matriz Bidimensional

<b>Python</b>	<b>C</b>
<pre>Notas=[[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0] 0]]  for coluna in range(10):     print("Aluno:",coluna+1)     Notas[0][coluna]=float(input("Digite a nota de P1 do aluno:"))     Notas[1][coluna]=float(input("Digite a nota de P2 do aluno:"))     Notas[2][coluna]=(Notas[0][coluna]+Notas[1][coluna])/2  print("Média dos Alunos:") for coluna in range(10):     print("Média do aluno ",coluna+1,Notas[2][coluna])     if Notas[2][coluna]&gt;=7:         print("Aluno aprovado")     else:         print("Aluno em recuperação")</pre>	<pre>#include &lt;stdio.h&gt; main() {     float Notas[3][10];     int coluna, aprovados;     for (coluna=0; coluna&lt;10; coluna++) {         printf("Digite a nota de P1:");         scanf("%f", &amp;Notas[0][coluna]);         printf("Digite a nota de P2:");         scanf("%f",&amp;Notas[1][coluna]);          Notas[2][coluna]=(Notas[0][coluna]+Notas[1][coluna])/2;     }     printf("Média dos alunos:");     for (coluna=0; coluna&lt;10; coluna++) {         printf("Média do aluno %d %.2f\n",         coluna+1,Notas[2][coluna]);         if (Notas[2][coluna]&gt;=7) {             printf("Aluno aprovado\n");         }         {             printf("Aluno em recuperação\n");         }     } }</pre>

Fonte: Os autores

## CAPÍTULO 5



# Funções

Ara que possamos resolver problemas complexos precisamos implementar algoritmos complexos. Para permitir um melhor entendimento dos algoritmos, problemas complexos podem ser divididos, utilizando-se a modularização do programa em pequenas partes, que podem ser denominadas de sub-rotinas, módulos ou subprogramas (SILVA et al., 2010). As sub-rotinas permitem organizar melhor a funcionalidade do programa, evitar a repetição de trechos de código (já que podem ser executadas diversas vezes), facilitar a depuração do código e tornar o código mais legível (SILVA et al., 2010). Na linguagem de programação Python as sub-rotinas são denominadas de funções e são declaradas por meio da palavra reservada `def`. A sintaxe para declarar uma função em Python é a seguinte:

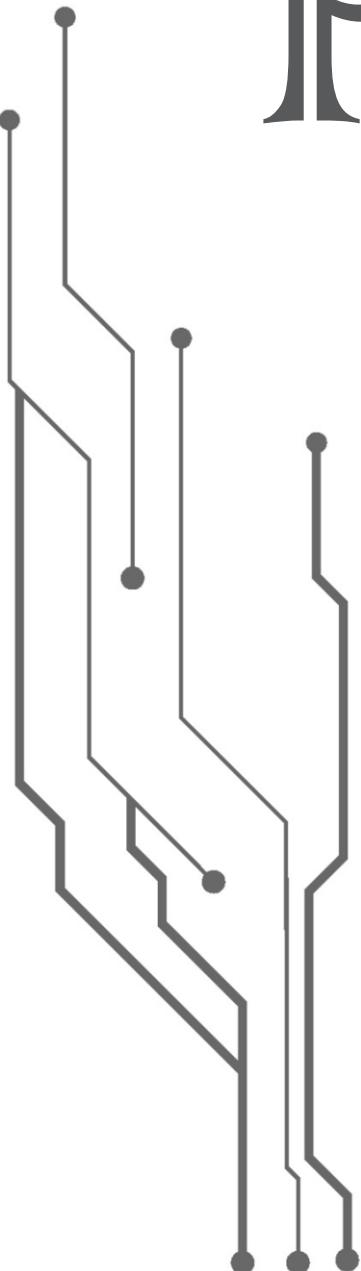
```
def nome_da_função(argumentos): """comentário – para que serve a função"""
```

Temos, de acordo com a sintaxe, que definir o `nome_da_função` e os parâmetros que deverão ser informados no momento em que a mesma será executada (quando "chamamos" a função em nosso código-fonte). Se não existirem parâmetros deve-se deixar os parênteses vazios (). Os comentários são opcionais e devem ser iniciados e terminados com 3 aspas duplas.

Por exemplo, vamos definir uma função que mostra uma mensagem na tela. Vamos chamar a função de `imprimir`:

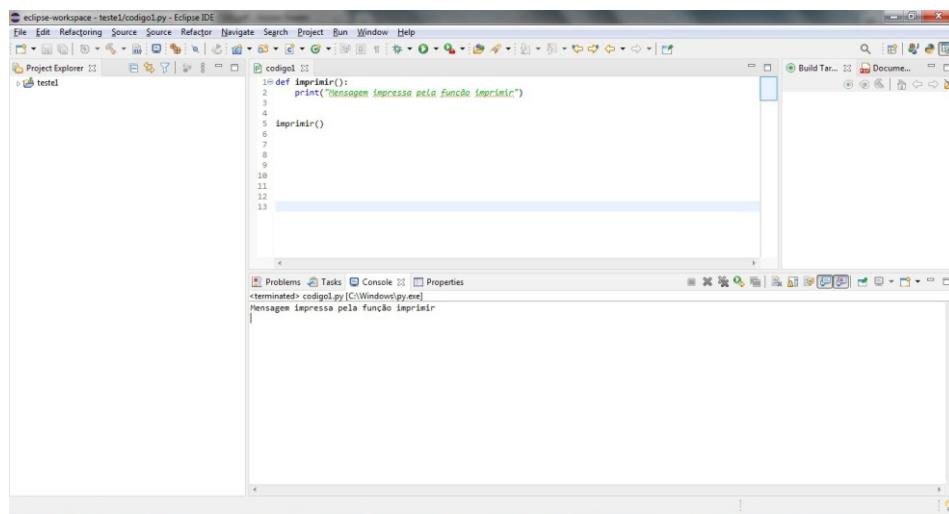
```
def imprimir():
    print("Mensagem impressa pela função imprimir")
```

P



Para utilizarmos (chamarmos) a função no nosso código-fonte, devemos referenciar o seu nome, com os parênteses: `imprimir()`. Vamos ver um exemplo na IDE Eclipse, como mostra a Figura 30. A função precisa estar definida antes de ser utilizada, ou seja, no código-fonte é preciso primeiro definir as funções (e o respectivo código de cada uma delas, após os dois pontos) para, posteriormente, podermos utilizá-las (chamá-las). Quando uma função é chamada ocorre um desvio de execução do programa para a linha/local onde a mesma está descrita. A partir de então, todos os comandos que fazem parte da função são executados. Ao final da execução o controle volta para o local onde previamente ocorreu o desvio, ou seja, o controle de execução volta para a rotina chamadora, que pode ser uma função ou a parte principal do programa (conhecido como programa principal, em C é representado pela função `main`).

Figura 30: Exemplo de Função em Python (Fonte: Os autores)



Vamos fazer um segundo exemplo de função, em que utilizaremos dois parâmetros: `nome` e `idade`:

```
def exemplo_funcao(nome, idade):
    print("Nome do usuário:", nome)
    print("Idade:", idade)
    if idade <= 18:
        print("Você é jovem!")
```

As variáveis `nome` e `idade` são os parâmetros da função. Para que possamos chamar esta função será preciso informar dois valores, um para a variável `nome` e outro para a variável `idade`. Quando definirmos os valores que serão “passados” para a função, esses valores são denominados de argumentos. Por exemplo:

```
exemplo_funcao("Maria", 15)
```

Maria e 15 são os argumentos que serão informados ao chamarmos a função `exemplo_funcao`.

Vamos ver o exemplo na IDE Eclipse, como mostra a Figura 31.

Figura 31: Exemplo de Função com Parâmetros (Fonte: Os autores)

The screenshot shows the Eclipse IDE interface. In the top-left corner, there's a 'Project Explorer' view showing a folder named 'test1'. Below it is a 'Console' view displaying the output of running the script. The code in the editor is:

```

1 def exemplo_funcao(nome, idade):
2     print("Nome do usuário:", nome)
3     print("Idade:", idade)
4     if idade<=18:
5         print("Você é jovem!")
6
7 exemplo_funcao("Maria", 15)
8
9
10
11
12
13
14
15

```

The console output shows:

```

Nome do usuário: Maria
Idade: 15
Você é jovem!

```

O Quadro 13 apresenta a comparação destes dois exemplos de funções em Python e em Linguagem C.

Quadro 13 – Comparação dos Exemplos de Funções

Python	C
<pre> #defineição da função def imprimir():     print("Mensagem impressa pela função imprimir")  # chamada da função imprimir() </pre>	<pre> #include &lt;stdio.h&gt;  void Imprimir() {     printf("Mensagem impressa pela função imprimir"); }  main() {     Imprimir(); } </pre>
<pre> #defineição da função def exemplo_funcao(nome, idade):     print("Nome do usuário:", nome)     print("Idade:", idade)     if idade&lt;=18:         print("Você é jovem!")  #chamada da função exemplo_funcao("Maria",15) </pre>	<pre> #include &lt;stdio.h&gt;  void Exemplo_funcao(char *Nome, int idade) {     printf("Nome do usuário: %s \n", Nome);     printf("Idade: %d \n", idade);     if (idade&lt;=18) {         printf("você é jovem! \n");     } }  main() {     Exemplo_funcao("Maria",15); } </pre>

Fonte: Os autores

Nos exemplos apresentados, as funções exibem os valores de saída, por meio de comandos de impressão na tela. Entretanto, uma função não precisa exibir, necessariamente, um valor de saída. A função pode receber parâmetros, processar e devolver um resultado para a rotina chamadora. Vamos utilizar o mesmo exemplo anterior e, ao invés de imprimirmos a mensagem de resposta, vamos devolver o resultado para a rotina chamadora (o que costumamos denominar de valor de retorno da função - return). Inserimos, dentro da função, a palavra reservada `return`, para definir o que a função deve retornar (ou devolver) para a rotina chamadora. No exemplo, retornamos a mensagem “Você é jovem”. Nesse caso, a rotina chamadora precisa estar adequada para receber o valor de retorno. O valor pode ser armazenado em uma variável ou impresso na tela. No exemplo da Figura 32, imprimimos o valor de retorno na tela, adicionando o comando `print` e, dentro dos parênteses, a chamada da função.

Figura 32: Exemplo de Função com Parâmetros e Retorno  
(Fonte: Os autores)

The screenshot shows the Eclipse IDE interface with a Python file named `codigo1.py` open in the editor. The code defines a function `exemplo_funcao` that prints user name and age, then returns a string based on age. It then calls this function with "Maria" and 15 as arguments and prints the result. The output window shows the user input and the program's response.

```

eclipse-workspace - teste1/codigo1.py - Eclipse IDE
File Edit Refactoring Source Source Refactor Navigate Search Project Run Window Help
Project Explorer Problems Tasks Console Properties
código1.py
1 def exemplo_funcao(nome, idade):
2     print("Nome do usuário:", nome)
3     print("Idade:", idade)
4     if idade <= 18:
5         return ("Você é jovem!")
6     else:
7         print(exemplo_funcao("Maria", 15))
8
9
10
11
12
13
14
15

```

```

Problems Tasks Console Properties
<terminated> codigo1.py [C:\Windows\py.exe]
Nome do usuário: Maria
Idade: 15
Você é jovem!

```

O Quadro 14 apresenta a comparação do exemplo da função desenvolvida na Figura 32 em Python e em Linguagem C.

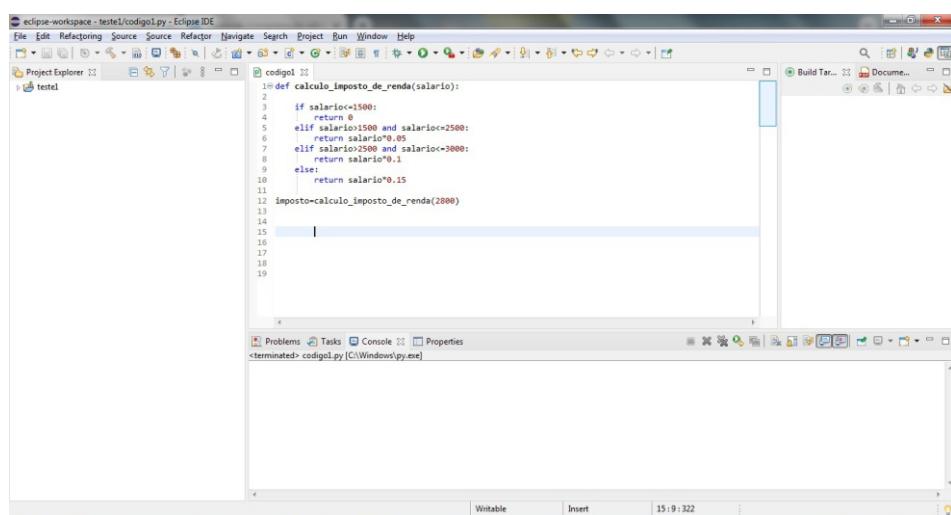
Quadro 14 – Comparação dos Exemplos de Funções

Python	C
<pre>#definição da função def exemplo_funcao(nome, idade):     print("Nome do usuário:", nome)     print("Idade:", idade)     if idade &lt;= 18:         return ("Você é jovem!")  #chamada da função print(exemplo_funcao("Maria", 15))</pre>	<pre>#include &lt;stdio.h&gt;  char* Exemplo_funcao(char *Nome, int idade) {     printf("Nome do usuário: %s\n", Nome);     printf("Idade: %d\n", idade);     if (idade &lt;= 18) {         return ("você é jovem\n");     } }  main() {     printf(Exemplo_funcao("Maria", 15)); }</pre>

Fonte: Os autores

Podemos, também, armazenar o valor de retorno em uma variável. Vamos fazer um exemplo de função que receba o valor do salário de uma pessoa e devolva o valor do Imposto de Renda a ser pago, de acordo com as alíquotas: até R\$1500,00 (isento), de R\$1501,00 a R\$2500,00 (5%), de R\$2501,00 a R\$3000,00 (10%) e acima de R\$3000,00 (15%). No exemplo da Figura 33, a rotina chamadora armazena o valor de retorno da função na variável `imposto`. Entretanto, em nosso exemplo, não estamos mostrando o valor na tela, apenas armazenando-o na variável.

Figura 33: Exemplo de Função com Retorno armazenado em uma variável (Fonte: Os autores)



The screenshot shows the Eclipse IDE interface with a Python file named `codigo1.py` open in the editor. The code defines a function `calculo_imposto_de_renda` that calculates income tax based on salary. The function uses nested if statements to determine the tax rate based on salary ranges. A call to this function is made with a salary of 2800, and the result is stored in the variable `imposto`. The code is as follows:

```

1 def calculo_imposto_de_renda(salario):
2     if salario<=1500:
3         return 0
4     elif salario>1500 and salario<=2500:
5         return salario*0.05
6     elif salario>2500 and salario<=3000:
7         return salario*0.1
8     else:
9         return salario*0.15
10
11 imposto=calculo_imposto_de_renda(2800)
12
13
14
15
16
17
18
19

```

The Eclipse interface includes a Project Explorer, a Problems view, and a Console view where the output of the script is shown.

O Quadro 15 apresenta a comparação destes dois exemplos de funções em Python e em Linguagem C.

Quadro 15 – Comparação dos Exemplos de Funções

<b>Python</b>	<b>C</b>
<pre> def calculo_imposto_de_renda(salario):      if salario&lt;=1500:         return 0     elif salario&gt;1500 and salario&lt;=2500:         return salario*0.05     elif salario&gt;2500 and salario&lt;=3000:         return salario*0.1     else:         return salario*0.15  imposto=calculo_imposto_de_renda(2800) </pre>	<pre> float imposto;  float calculo_imposto_de_renda(float salario) {     if (salario &lt;=1500)         return 0;     if (salario&gt;1500 and salario&lt;2500)         return salario*0.05;     if (salario&gt;2500 and salario&lt;=3000)         return salario*0.1;     if (salario&gt;3000)         return(salario*0.15); }  main () {     imposto=calculo_imposto_de_renda(2800); } </pre>

Fonte: Os autores

Quando incluímos funções em nossos programas precisamos estudar o escopo das variáveis, que pode ser global ou local. Uma variável definida no programa principal (rotina *main*, por exemplo, é uma variável local). Por sua vez, uma variável definida dentro de uma função, é uma variável local. No exemplo da Figura 33, a variável *imposto* é global, pois foi definida fora da função. Sendo assim, ela pode ser acessada em todo o programa, até mesmo em outras funções. Veja o exemplo da Figura 34.

Figura 34: Exemplo de Variável Global (Fonte: Os autores)

```

eclipse-workspace - teste1/codigol.py - Eclipse IDE
File Edit Refactoring Source Source Refactor Navigate Search Project Run Window Help
Project Explorer 23 Build Tar... Document...
codigol.py
1#def calculo_imposto_de_renda(salario):
2    if salario<=1500:
3        return 0
4    elif salario>1500 and salario<=2500:
5        return salario*0.05
6    elif salario>2500 and salario<=3000:
7        return salario*0.1
8    else:
9        return salario*0.15
10
11
12def exemplo_variavel_global():
13    print("Valor do imposto a pagar:", imposto)
14
15
16imposto=calculo_imposto_de_renda(2800)
17exemplo_variavel_global()
18
19
20
21
22

```

Problems Tasks Console Properties

<terminated> codigol.py [C:\Windows\py.exe]

Valor do imposto a pagar: 280.0

Vemos, no exemplo da Figura 34, que a variável *imposto* foi utilizada na função *exemplo\_variavel\_global*, sem que precisasse ser definida dentro da função e nem mesmo seu valor tivesse sido passado por parâmetro. Apesar dessa possibilidade existir, por questões de legibilidade do código-fonte e, também, de segurança (para que uma função não modifique o valor de uma variável global - o que pode afetar todo o programa), não recomendamos o uso de variáveis globais dentro das funções.

O quadro 16 apresenta o código da Figura 34 em Python e na Linguagem C.

Quadro 16 – Comparação dos Exemplos de Funções

<i>Python</i>	<i>C</i>
<pre> def calculo_imposto_de_renda(salario):     if salario&lt;=1500:         return 0     elif salario&gt;1500 and salario&lt;=2500:         return salario*0.05     elif salario&gt;2500 and salario&lt;=3000:         return salario*0.1     else:         return salario*0.15  def exemplo_variavel_global():     print("Valor do imposto a pagar:", imposto)  imposto=calculo_imposto_de_renda(2800) exemplo_variavel_global() </pre>	<pre> #include &lt;stdio.h&gt; float imposto;  float calculo_imposto_de_renda(float salario) {     if (salario &lt;= 1500)         return 0;     if (salario&gt;1500 and salario&lt;2500)         return salario*0.05;     if (salario&gt;2500 and salario&lt;=3000)         return salario*0.1;     if (salario&gt;3000)         return(salario*0.15); }  void exemplo_variavel_global() {     printf("Valor do imposto a pagar: %f",imposto); }  main () {     imposto=calculo_imposto_de_renda(2800);     exemplo_variavel_global(); } </pre>

Fonte: Os autores

Vamos ver o código do exemplo anterior (Figura 34 e Quadro 16) utilizando a passagem de parâmetros ao invés de uma variável global. O Quadro 17 mostra o exemplo do código-fonte em Python e na Linguagem C. No exemplo, utilizando o nome `imposto` (que é uma variável global) como nome do parâmetro da função. Entretanto, apesar de serem nomes iguais, ao passarmos `imposto` como parâmetro, estamos utilizando a passagem de parâmetros por valor. Sendo assim, a função recebe uma cópia do valor e armazena-o em outra posição da memória. Se fizermos alguma modificação no valor da variável `imposto`, dentro da função, isso só acontecerá no escopo local (dentro da função).

Quadro 17 – Comparação dos Exemplos de Funções

<b>Python</b>	<b>C</b>
<pre>def calculo_imposto_de_renda(salario):     if salario&lt;=1500:         return 0     elif salario&gt;1500 and salario&lt;=2500:         return salario*0.05     elif salario&gt;2500 and salario&lt;=3000:         return salario*0.1     else:         return salario*0.15  def exemplo_variavel_global(imposto):     print("Valor do imposto a pagar")     imposto  imposto=calculo_imposto_de_renda(2800) exemplo_variavel_global(imposto)</pre>	<pre>#include &lt;stdio.h&gt; float imposto;  float calculo_imposto_de_renda(float salario) {     if (salario &lt;=1500)         return 0;     if (salario&gt;1500 and salario&lt;2500)         return salario*0.05;     if (salario&gt;2500 and salario&lt;=3000)         return salario*0.1;     if (salario&gt;3000)         return(salario*0.15); }  void exemplo_variavel_global(float imposto) {     printf("Valor do imposto a pagar: %f",imposto) }  main () {     imposto=calculo_imposto_de_renda(2800);     exemplo_variavel_global(imposto); }</pre>

Fonte: Os autores

Utilizando o mesmo exemplo, vamos adicionar uma linha de código dentro da segunda função. Vamos alterar o valor do parâmetro `imposto`, multiplicando-o por dois. Ao executar o código (Figura 35) vemos que o valor aparece alterado dentro da função (escopo local) mas, ao voltar a execução para a rotina chamadora (escopo global), o valor não foi modificado.

O Quadro 18 apresenta o código-fonte do exemplo da Figura 35, comparado entre as linguagens Python e C.

Figura 35: Exemplo com escopo local e global (Fonte: Os autores)

The screenshot shows the Eclipse IDE interface with a Python file named 'codigol.py' open in the editor. The code defines two functions: 'calculo\_imposto\_de\_renda' and 'exemplo\_variavel\_global'. The 'calculo\_imposto\_de\_renda' function calculates tax based on salary. The 'exemplo\_variavel\_global' function prints the tax amount to be paid, updates it by 200, and then prints it again after the function call. The output window shows the results of running the script.

```

eclipse-workspace - teste1/codigol.py - Eclipse IDE
File Edit Refactoring Source Search Project Run Window Help
Project Explorer Build Target Document
codigol.py
> testel
1 def calculo_imposto_de_renda(salario):
2     if salario<=1500:
3         return 0
4     elif salario>1500 and salario<=2500:
5         return salario*0.05
6     elif salario>2500 and salario<=3000:
7         return salario*0.1
8     else:
9         return salario*0.15
10
11 def exemplo_variavel_global(imposto):
12     print("Valor do imposto a pagar:", imposto)
13     imposto+=imposto*2
14     print("Valor do imposto atualizado no escopo local:",imposto)
15
16     imposto=calculo_imposto_de_renda(2800)
17     print("Valor do imposto no escopo global:",imposto)
18     exemplo_variavel_global(imposto)
19     print("Valor do imposto após a execução da função:",imposto)
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
928
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
958
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158

```

Figura 36: Exemplo utilizando uma variável para armazenar o valor de retorno da função (Fonte: Os autores)

The screenshot shows the Eclipse IDE interface. In the top window, a Python file named 'codigol.py' is open, containing the following code:

```

1 def calculo_imposto_de_renda(salario):
2     if salario<=1500:
3         imposto=0
4     elif salario>1500 and salario<=2500:
5         imposto=salario*0.05
6     elif salario>2500 and salario<=3000:
7         imposto=salario*0.1
8     else:
9         imposto=salario*0.15
10    return imposto
11
12
13 imposto=calculo_imposto_de_renda(2800)
14 print("Valor do imposto:",imposto)
15
16
17
18
19
20
21
22
23

```

In the bottom window, the console output shows the result of running the code:

```

Valor do imposto: 280.0

```

No Quadro 19 é possível verificarmos a comparação do código-fonte do exemplo da Figura 36, entre Python e C.

Quadro 19 – Comparação dos Exemplos com uma variável para retornar o resultado da função

<b>Python</b>	<b>C</b>
<pre> def calculo_imposto_de_renda(salario):     if salario&lt;=1500:         imposto=0     elif salario&gt;1500 and salario&lt;=2500:         imposto=salario*0.05     elif salario&gt;2500 and salario&lt;=3000:         imposto=salario*0.1     else:         imposto=salario*0.15     return imposto  imposto=calculo_imposto_de_renda(2800) print("Valor do imposto:",imposto) </pre>	<pre> #include &lt;stdio.h&gt; float imposto;  float calculo_imposto_de_renda(float salario) {     float imposto;     if (salario &lt;=1500)         imposto=0;     if (salario&gt;1500 and salario&lt;2500)         imposto=salario*0.05;     if (salario&gt;2500 and salario&lt;=3000)         imposto=salario*0.1;     if (salario&gt;3000)         imposto=salario*0.15;     return imposto; }  main () {     imposto=calculo_imposto_de_renda(2800);     printf("Valor do imposto: %f",imposto); } </pre>

Fonte: Os autores



# BIBLIOGRAFIA

## REFERÊNCIAS BIBLIOGRÁFICAS

FALKEMBACH, Gilse Morgental; SILVEIRA, Sidnei Renato. Algoritmos e Programação I. Canoas: ULBRA, 2005. Caderno Universitário.

MATTHES, Eric. Curso Intensivo de Python: uma introdução prática e baseada em projetos de programação. São Paulo: Novatec, 2016.

PARREIRA, Fábio José; SILVEIRA, Sidnei Renato; BERTOLINI, Cristiano; SEVERO, Rosane Beatriz. Introdução a Algoritmos. Santa Maria: UAB/NTE/UFSM, 2017. Disponível em: [https://repositorio.ufsm.br/bitstream/handle/1/15820/Licenciatura\\_Computacao\\_introducaoalgoritmos.pdf?sequence=1&isAllowed=y](https://repositorio.ufsm.br/bitstream/handle/1/15820/Licenciatura_Computacao_introducaoalgoritmos.pdf?sequence=1&isAllowed=y). Acesso em 21 out. 2019.

PYTHON.ORG. Welcome to Python.org. Disponível em: <http://python.org>. Acesso em 21 out. 2019.

ROSSUM, G. Tutorial Python. Disponível em: [https://wiki.python.org.br/Tutorial\\_Python](https://wiki.python.org.br/Tutorial_Python). Acesso em 20 out. 2019.

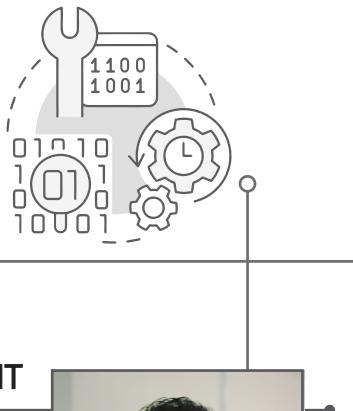
SILVA, Isabel Cristina Siqueira; FALKEMBACH, Gilse Morgental; SILVEIRA, Sidnei Renato. Algoritmos e Programação em Linguagem C. Porto Alegre: UniRitter, 2010.

SILVEIRA, Sidnei Renato; DE VIT; Antônio Rodrigo Delepine; BERTOLINI, Cristiano; PARREIRA, Fábio José; CUNHA; Guilherme Bernardino; BIGOLIN, Nara Martini. Paradigmas de Programação: uma introdução. Belo Horizonte: Synapse, 2021. Disponível em: [https://www.editorasynapse.org/wp-content/uploads/2021/03/paradigmas\\_programacao\\_uma\\_introducao\\_V0.pdf](https://www.editorasynapse.org/wp-content/uploads/2021/03/paradigmas_programacao_uma_introducao_V0.pdf). Acesso em 30 mar. 2021.

# INTRODUÇÃO À PROGRAMAÇÃO E ESTRUTURAS DE DADOS FUNDAMENTAIS COM PYTHON PARA PROGRAMADORES C



## APRESENTAÇÃO DOS AUTORES



## APRESENTAÇÃO DOS AUTORES

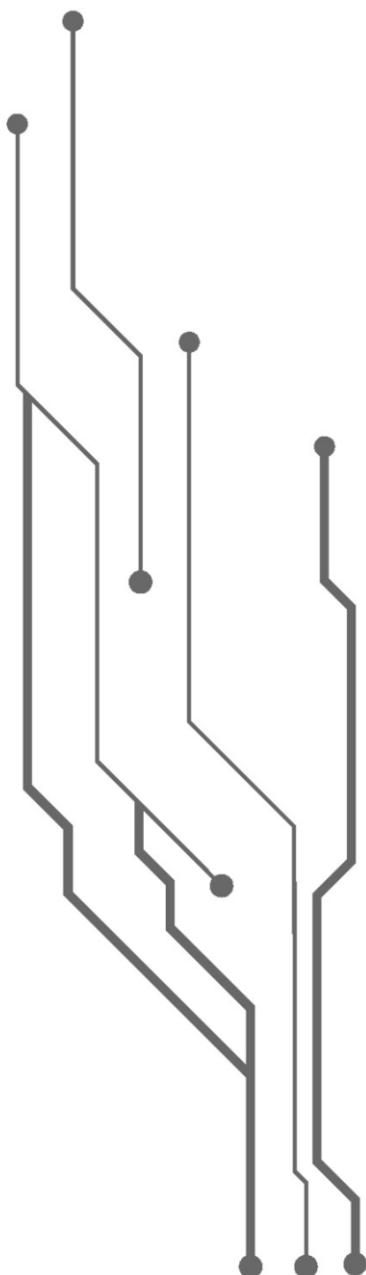
### ANTÔNIO RODRIGO DELEPIANE DE VIT

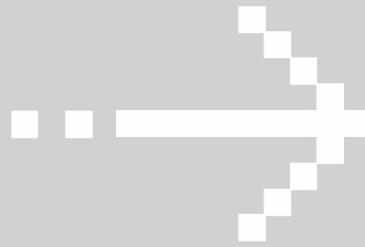
Professor Adjunto do Departamento de Tecnologia da Informação da UFSM (Universidade Federal de Santa Maria) - Campus Frederico Westphalen/RS. Doutor em Ciência da Computação pela PUC-RS (Pontifícia Universidade Católica do Rio Grande do Sul). Mestre em Ciência da Computação pela UFRGS (Universidade Federal do Rio Grande do Sul) e Bacharel em Informática pela UNIJUÍ.



### SIDNEI RENATO SILVEIRA

Professor Associado do Departamento de Tecnologia da Informação da UFSM (Universidade Federal de Santa Maria) - Campus Frederico Westphalen/RS. Doutor em Ciência da Computação pela UFRGS (Universidade Federal do Rio Grande do Sul). Mestre em Ciência da Computação pela UFRGS. Especialista em Gestão Educacional pelo SENAC. Especialista em Administração e Planejamento para Docentes pela ULBRA. Bacharel em Informática pela ULBRA.





<https://www.facebook.com/Synapse-Editora-111777697257115>



<https://www.instagram.com/synapseeditora>



<https://www.linkedin.com/in/synapse-editora-compartilhando-conhecimento/>



31 98264-1586



[editorasynapse@gmail.com](mailto:editorasynapse@gmail.com)



Compartilhando conhecimento