**Oficina de Programação**

**Igor Barreto**

**José Augusto**

Sumário

[1. O que é a Computação 3](#_Toc16482824)

[2. Entrada/saída de dados (Input/Output) 3](#_Toc16482825)

[3. Tipos de Dados 4](#_Toc16482826)

[1. Operadores 4](#_Toc16482827)

[1.1 Operadores Aritméticos 4](#_Toc16482828)

[1.2 Operadores Lógicos 5](#_Toc16482829)

[1.3 Operadores relacionais 5](#_Toc16482830)

[4.5 Ordem entre Operadores 6](#_Toc16482831)

[4. Python 6](#_Toc16482832)

[1.4 Variáveis 6](#_Toc16482833)

[1.5 Entrada de dados 6](#_Toc16482834)

[1.6 Saída de dados 7](#_Toc16482835)

# O que é a Computação

A computação pode ser definida como a busca de uma solução para um problema a partir de entradas (inputs) e depois tem seus resultados (outputs). Os resultados são feitos através de um algoritmo.

Exemplos: rádio, jogos, videogame, carro autônomo, etc.

O algoritmo é uma sequência de passos finitos e bem definidos para a resolução do problema. Ou seja, é uma sequência de coisas que o computador deve fazer para que, com as entradas que ele tem, ele consiga gerar o resultado esperado.

O computador é rápido, porém, não é inteligente, por isso devemos ensinar ele a fazer o processo. Para conversarmos com ele, utilizamos linguagem de programação.

Exemplos: Python, Java, C, Assembly, etc.

Através disso, podemos utilizar a linguagem de programação para desenvolver algoritmos que podem ser aplicados em:

* Inteligência Artificial: ensinar robôs a falarem, carro autônomos, traçar melhor rota do GPS, bots em jogos.
* Processamento de Imagens: reconhecimento de placas, reconhecimento de pessoas.
* Desenvolvimento de sites: Facebook, Google, Tecmundo.

E isso pode ser aplicado em diversas áreas:

* Saúde: programa para agendamento de consultas.
* Engenharia: programa para desenhar planta de construção, desenhar circuitos elétricos.

É interessante aprender sobre a computação, pois ajuda a pensar de maneira diferente para resolver um mesmo problema, desenvolve o raciocínio lógico. O raciocínio lógico é uma forma de estruturar o pensamento, ou seja, organizar o pensamento, quebrando e fazendo uma sequência de etapas a serem seguidas para a resolução de um determinado problema.

# Entrada/saída de dados (Input/Output)

* Entrada: são dispositivos que enviam dados, ou comandos, para o computador realizar alguma ação.
  + Ex: mouse, teclado, controle remoto.
* Saída: são dispositivos que mostram os resultados de uma ação.
  + Ex: monitor, caixa de som, impressora.

Pen drive, HD, cartão de memória não são considerados como dispositivos de entrada ou de saída, e sim, como dispositivos de armazenamento, pois a principal função deles são de armazenar dados.

# Tipos de Dados

Cada tipo de dado tem uma finalidade, por exemplo: escrever uma frase, mostrar um salário, mostrar se a operação é verdadeira, por isso existe várias maneiras de representar um dado. No python, são aceitos os seguintes tipos de dados:

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Python** | **Significado** |
| Inteiro | int | Usado para números inteiros. Ex: 4, 0, -2 |
| Real | float | Utilizado para números quebrados. Ex: -6.7, 0.2, 3.7 |
| Texto | str (string) | Representa palavras ou texto. ATENÇÃO o texto DEVE ser colocados entre aspas ( ‘ ‘ ).Ex: ‘casa’, ‘eu sou uma frase’ |
| Booleano | bool | Possui apenas dois estados: verdadeiro ou falso (True / False) |
| Lista | list | É utilizados para armazenar vários tipos de variáveis:  Compras = [‘açucar’,’leite’,’farinha’,’ovos’] |
| Tupla | tuple | É semelhante a uma lista porém ela é imutável, ou sjea depois que é criada ela não pode ser modificada. Não pode adicionar ou remover itens de dentro dela:  Pos\_time\_futebol(‘goleiro’,’zagueiro’,’atacante’,’meia’) |
| Dicionário | dict | Utiliza-se esta representação de dado quando precisamos armazenar informações complexas e precisamos nomear estas informações.  Pessoa= {nome:’Igor’, idade=26, peso: 75.00, altura: 1.77, sexo=’M’} |

# Operadores

## Operadores Aritméticos

Os operadores aritméticos são os operadores matemáticos no qual estamos acostumados. A representação no Python:

|  |  |
| --- | --- |
| Operador | Significado |
| + | Soma. Ex:  1 + 1 = 2 |
| - | Subtração. Ex:  3 – 2 = 1 |
| \* | Multiplicação. Ex:  2 \* 3 = 6 |
| / | Divisão. Ex:  12 / 3 = 4 |
| // | Parte inteira da divisão. Ex: 11 // 3 = 3 |
| % | Resto da divisão. Ex:  11 % 3 = 2 |
| \*\* | Potência. Ex:  2 \*\* 3 = 2³ = 8 |

Para resolvermos os operadores aritméticos, devemos seguir a seguinte sequência de resolução:

|  |  |
| --- | --- |
| 1º | ( ) |
| 2º | \*\* |
| 3º | \*, /, //, % |
| 4º | +, - |

Primeiro resolvemos todas operações que estão entre parênteses, depois as potencias, e após, resolvemos multiplicações, divisões, inteiro da divisão e resto de divisão, e, por último, fazemos as somas e subtrações. Na 3ª e 4ª sequência, como temos vários operadores com a mesma prioridade, a resolução é conforme é feito a leitura.

## Operadores Lógicos

Operadores lógicos são comparadores de verdadeiro e falsos. Cada tipo de operação retornará um resultado, conforme as tabelas:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **E** | | |  | **OU** | | |
| **A** | **B** | **RESULTADO** |  | **A** | **B** | **RESULTADO** |
| V | V | V |  | V | V | V |
| V | F | F |  | V | F | V |
| F | V | F |  | F | V | V |
| F | F | F |  | F | F | F |

|  |  |
| --- | --- |
| **NOT (~ , ¬)** | |
| **A** | **RESULTADO** |
| V | F |
| F | V |

Uma forma fácil de lembrar os operadores lógicos é, na operação E, só vai ser VERDADEIRO se as duas variáveis forem VERDADEIRAS. Na operação OU só irá ser FALSO se ambas variáveis forem FALSAS. E na operação NOT basta inverter.

Para a resolução dos operadores lógicos, devemos seguir a seguinte ordem:

|  |  |
| --- | --- |
| 1º | ( ) |
| 2º | NOT |
| 3º | E |
| 4º | OU |

## Operadores relacionais

Os operadores relacionais são utilizados para fazer comparação de valores, o resultado dessas operações será VERDADEIRO ou FALSO. No Python é aceito os seguintes operadores relacionais:

|  |  |
| --- | --- |
| **OPERADOR** | **SIGNIFICADO** |
| > | A é maior que B |
| < | A é menor que B |
| >= | A é maior ou igual a B |
| <= | A é menor ou igual a B |
| == | A é igual a B |
| =! | A é diferente que B |

A ordem de resolução é conforme se lê.

## Ordem entre Operadores

Quando houver vários tipos de operadores para serem resolvidos, deve-se seguir a seguinte sequência de resolução:

|  |  |
| --- | --- |
| 1º | Operadores aritméticos |
| 2º | Operadores relacionais |
| 3º | Operadores lógicos |

# Python

## Variáveis

As variáveis são uma identificação de um local na memória do computador em que você irá salvar algum tipo de dado. Ela serve para facilitar o programador para que ele consiga reutilizar a informação que está contida na variável de uma maneira mais fácil.

O nome da variável não pode ser inicializado com números ou conter caracteres especiais (“, Ç, /, =, !, @, #, $, %, &, \, (), [], {}, ~, ^, ´), exceto o underline (“ \_ “),

Ex:

1casa, teste!, “nome”, a=a, @email, programação, etc.

Exemplos de variáveis que podem ser escritas:

casa1, Nome, testePrograma, \_programacao, etc.

Além disso, ela não pode ser também uma palavra reservada do python, pois essas palavrar são utilizadas para o programa realizar as ações que ele precisa fazer. As palavrar reservadas são:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| and | as | assert | break | class | continue | def | del | elif | else |
| exceto | exec | finally | for | from | global | if | import | in | is |
| lambda | nonlocal | not | or | pass | raize | return | try | while | with |
| yield | True | False | None |  |  |  |  |  |  |

## Entrada de dado

## Entrada de dados

Para salvar algum dado no seu programa, utiliza-se variáveis. As variáveis podem ser pré-definidas ou pode-se utilizar o comando “input” para solicitar ao usuário que irá usar o seu programa a entrar com os dados solicitados. Exemplos de variáveis pré-definidas:

* fruta = “laranja”
* quantidade = 15
* fazSuco = True

Para que o usuário entre com algum dado, usasse o comando input, que contém a seguinte estrutura:

variável = input (“sua mensagem”)

* variável = O nome da sua variável
* “sua mensagem” = Mensagem que aparecerá na tela para identificar o que usuário deve inserir

Exemplos:

* nome = input (“Insira seu nome: “)
* profissao = input (“Qual a sua profissão? “)

Quando precisamos armazenar algum número como o preço de algum produto ou a quantidade dele precisamos utilizar algumas funções extras, veja o exemplo a seguir:

Exemplo:

Precisamos que informe quantas pessoas possui em sala de aula para dividirmos em em equipes iguais.

Neste caso queremos saber quantas pessoas tem, isso é um número inteiro então precisamos escrever desta maneira:

numeroPessoasNaTurma = int(input(‘Quantas pessoas tem na turama’))

E agora precisamos saber a nota das pessoas, então normalmente uma nota é um número com casas decimais, então utilizamos uma representação de números flutuantes que fica da seguinte forma:

notadoAluno = float(input(‘Qual a nota do aluno?’))

\*QUANDO VAI CRIAR UMA VARIÁVEL PRECISA PENSAR EM QUE TIPO DE DADO ELA É, SE VOCÊ VAI APLICAR ALGUMA FUNÇÃO MATEMÁTICA, ISSO INFLUNCIARÁ NA ESCOLHA DO TIPO DE DADO

## Saída de dados

Para mostrar uma mensagem ou um resultado na tela, utiliza-se o comando “print”. Pode utilizar para mostrar mensagens, valor de uma variável ou ambos. Exemplos:

* print (“Uma mensagem está aqui”)
* print (variável)
* print (“Olá, seja bem-vindo “ + nome)

Atualmente utiliza-se esta maneira para representar uma frase que contenha textos e variáveis, pois sua sintaxe, maneira de escrever, é mais simples e mais legível:

* print(f‘O valor da some entre {numero1} e {numero2} é {resultado}’)

## Condicionais

Os condicionais servem para desviar um fluxo no programa, quando precisamos tomar um caminho se a alguma condição for atendida, veja o exemplo a seguir para uma melhor compreensão

Por exemplo você está no trânsito indo para a casa de um amigo(a), porém você nunca foi lá então está utilizando o GPS, neste momento você chega a uma bifurcação, o GPS indica que o lado direito está congestionado devido a um acidente e o lado esquerdo está livre para passar, qual você escolheria?



Espero que você escolha o lado que te leve o mais rápido possível a casa do seu amigo(a).

Este exemplo foi para ilustrar o que é um condicional.

## If/else

Em uma linguagem de programação isso é representado pelo conjunto de palavras if, else e elif da seguinte maneira:

A estrutura de um condicional é a seguinte:

if [CONDIÇÃO] :

//CÓDIGO

elif [CONDIÇÃO]:

//CÓDIGO

else:

//CÓDIGO

Para que rode o que estiver dentro do if, por exemplo a condição tem que ser verdadeira, se não o código não será executado pelo programa.

Outro ponto importante é a questão tabulação, quando colocamos um código dentro do if para ser avaliado precisamos colocar uma tabulação que no teclado é feito pela tecla TAB C:\Users\igorb\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\A496028D.tmp

Abaixo segue um exemplo de como podemos aplicar o condicional e a sua importancia

ladoEsquerdoCongestinado= True

if ladoEsquerdoCongestinado:

print(‘seguir pela direita’)

else:

print(‘seguir pela esquerda’)

## Laços de repetição

Os laços de repetição tem um papel importante na programação, por causa deles não precisa escrever várias vezes o mesmo código ou sabermos a hora de interromper, por exemplo, um personagem que anda 2 quadrados a cada vez que envia um sinal para se movimentar, e nesse caso ele precisa se deslocar 100 quadrados, se não tivesse o laço teria que realizar 50 instruções para ele atingir o objetivo



Sem laço

Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar() Andar()

Com laço

ENQUANTO não chegar no final:

Andar()

### For

For é um tipo de laço , normalmente utilizamos ele quando sabemos quantas vezes precisamos executar um bloco de código, ou seja, ele obriga a dizer quantas interações precisa ser realizada. Um exemplo, armazenar a nota de 10 alunos, ou calcular a soma de todos os valores de uma lista.

Estrutura do laço For:

For [variável] in [objeto iterável]:

// código

Exemplo:

Precisa-se mostrar todos os valores de 1 até 100 na tela.

For i in range(101):

print(i)

IMPORTANTE: Como no condicional, precisa ficar atento a tabulação, por que quando precisar colocar um bloco dentro do laço adicione uma tabulação.

### While

O While segue a mesma ideia do For, mas com uma pequena diferença, ele é utilizado para rodar um bloco de código enquanto uma condição for verdadeira.

Estrutra:

While [CONDIÇÃO] :

//CÓDIGO

Exemplo:

Precisa fazer a soma do valor de todos os produtos feitos em uma compra.

temMais = True:

valorTotal =0.0

while temMais :

valorTotal+= float(input(‘Qual o valor do próximo produto?’))

acabou = input(‘Tem mais produtos? [S/N]’)

if acabou != ‘S’ :

temMais = False

Neste caso, percebe-se que o programa só ira acabar quando entrarmos com o valor ‘S’ quando pergunta ‘se tem mais produtos?’

### Funções

### Strings

### Listas /Tuplas/Dicionários