

Programação em Python



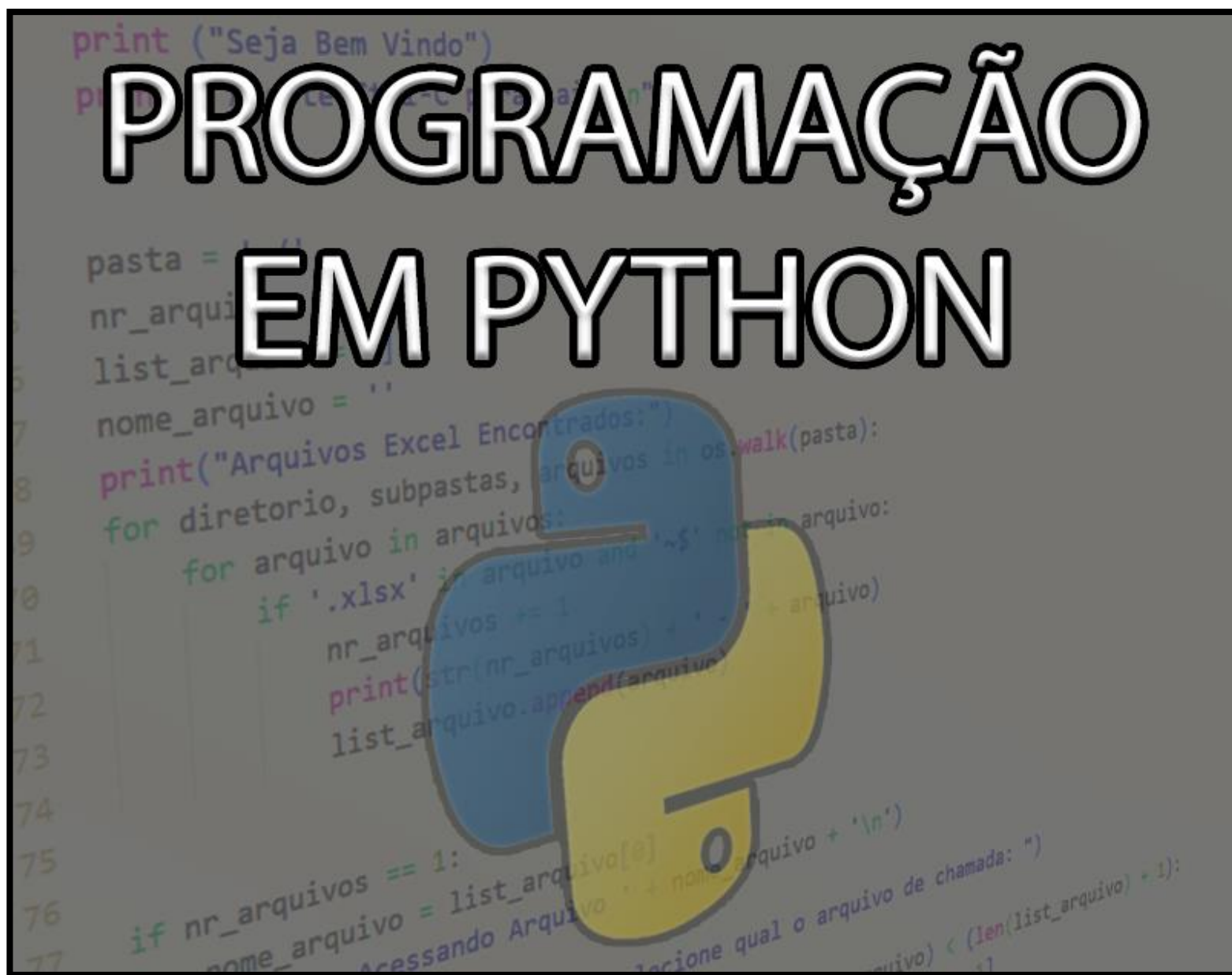
Prof. Daniel Santos

daniel.sampaio@sp.senai.br

Senai Roberto Simonsen

R. Monsenhor Andrade, 298 - Brás, São Paulo - SP

PROGRAMAÇÃO EM PYTHON



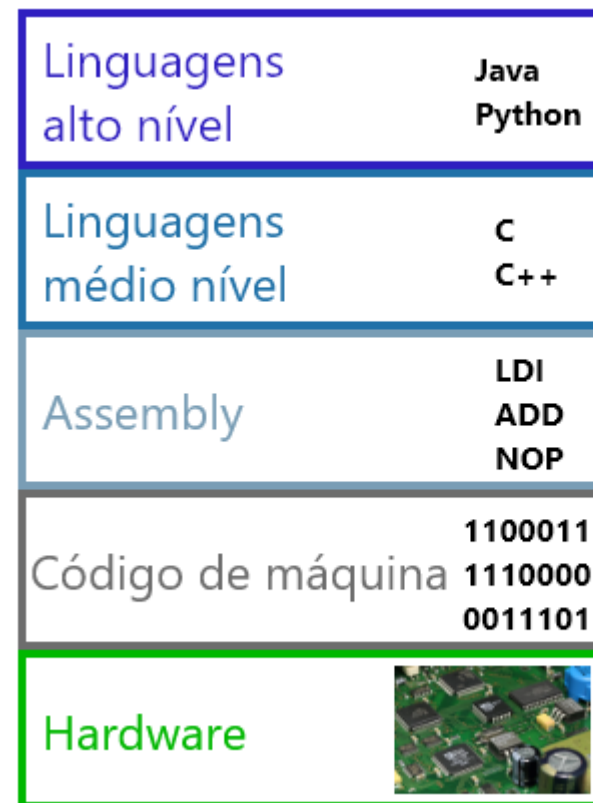
Prof. Daniel Santos

O que é Python?

Python é uma linguagem de programação de alto nível — ou **High Level Language** —, dinâmica, interpretada, modular, multiplataforma e orientada a objetos.



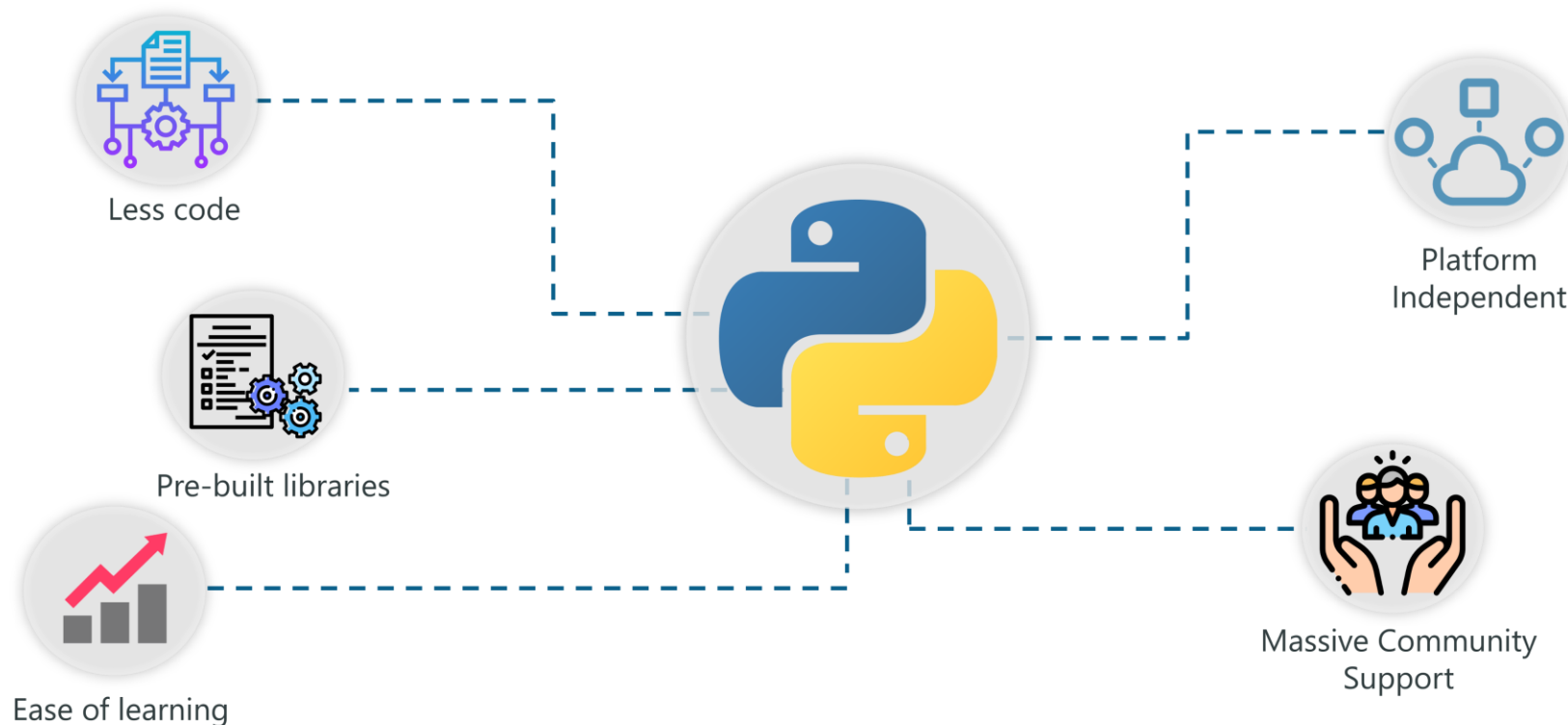
| Linguagem de Máquina | Linguagem de Baixo Nível | Linguagem de Alto Nível |
|----------------------|--------------------------|-------------------------|
| 0010 0001 1110 | LOAD R1, val1 | val2 = val1 + val2 |
| 0010 0010 1111 | LOAD R2, val2 | |
| 0001 0001 0010 | ADD R1, R2 | |
| 0011 0001 1111 | STORE R1, val2 | |



Maior nível de abstração

O que é Python?






Um de seus maiores atrativos é possuir um **grande número de bibliotecas**, nativas e de terceiros, tornando-a muito difundida e útil em uma grande variedade de setores dentro de **desenvolvimento web**, e também em áreas como **análise de dados**, **machine learning** e **Inteligência Artificial**.



Por que aprender Python?



- É uma das linguagens de programação mais procurada na atualidade.
- Tem um vasto repositório de bibliotecas para facilitar a programação de projetos mais complexos
- Esta linguagem pode ser aplicada em diversos campos, desde **ciência da computação**, **machine learning**, **desenvolvimento web** até para área de **automação e domótica**.
- A comunidade é bem ativa, então se houver alguma dúvida específica é possível pedir auxílio para outros desenvolvedores nos fóruns (ex: <https://groups.google.com/g/python-brasil>)

| Feb 2023 | Feb 2022 | Change | Programming Language | | Ratings | Change |
|----------|----------|--------|---|--------|---------|--------|
| 1 | 1 | |  | Python | 15.49% | +0.16% |
| 2 | 2 | |  | C | 15.39% | +1.31% |
| 3 | 4 | ▲ |  | C++ | 13.94% | +5.93% |
| 4 | 3 | ▼ |  | Java | 13.21% | +1.07% |
| 5 | 5 | |  | C# | 6.38% | +1.01% |

Raciocínio lógico



- Para aprender qualquer linguagem de programação é primordial ter **raciocínio lógico**
- Ter **raciocínio lógico** é pensar seguindo uma **sequência lógica** de instruções para executar alguma tarefa, isto não precisa ser necessariamente aplicado a programação.

Exemplo: **Fazer um bolo.** Para fazer um bolo é necessário seguir uma sequência de tarefas ordenadas.

```
// RECEITA DE BOLO COMUM DE OVOS
INÍCIO
Passo 1: Separar os ingredientes
Ingredientes:
2 ovos;
3 xícaras de farinha de trigo;
1 e ½ colher de fermento;
¾ xícara de leite.
1/2 xícaras de açúcar;
250g de manteiga;

Modo de preparo:
Passo 2: Aqueça o forno a 180 graus;
Passo 3: Quebre os ovos e separe as claras da gema;
Passo 4: Bata as claras em neve e as deixe separadas;
Passo 5: Em uma vasilha, bata o açúcar, a manteiga e as gemas;
Passo 6: Misture a farinha e o leite;
Passo 7: Bata bem, até ficar bem homogêneo;
Passo 8: Acrescente o fermento;
Passo 9: Adicione as claras em neve e mexa cuidadosamente;
Passo 10: Unte uma forma com manteiga e farinha de trigo.
Passo 11: Coloque a massa na forma untada
Passo 12: Leve ao forno médio para assar por aproximadamente 35 minutos ou até que, ao
    espetar um palito, esse saia seco;
Passo 13: Após assado, desligue o forno e deixe o bolo esfriar;
Passo 14: Desenforme e saboreie.
FIM
```

O nome dado a esta sequência de passos para executar uma tarefa é **algoritmo**.

Algoritmo

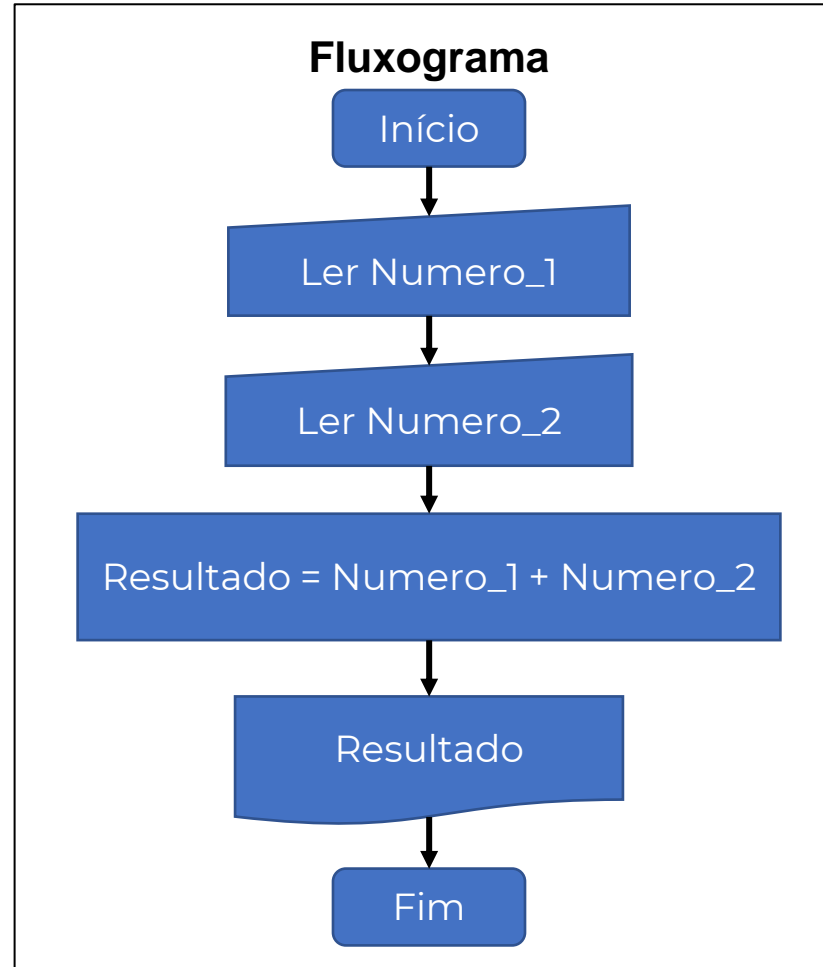


Podemos representar um algoritmo de algumas maneiras diferentes:

Descrição Narrativa

Passo 1: Receber o primeiro número
Passo 2: Receber o segundo número
Passo 3: Somar os dois números
Passo 4: Exibir resultado

Fluxograma



Pseudocódigo (PORTUGOL)

```
Declare Numero_1, Numero_2, Resultado;  
Leia Numero_1, Numero_2;  
Resultado = Numero_1 + Numero_2;  
Escreva Resultado.
```

Comparação Python x JAVA



JAVA

```
public class welcome {  
    public static void main (String args[ ] ) {  
        System.out.println (" Ola mundo!");  
    }  
}
```

PYTHON

```
print ( ' Ola mundo!' )
```


Linguagem Python

Assim como aprender um idioma, para aprender uma linguagem de programação é necessário entender a **sintaxe**, ou seja, como ela é escrita:

- A linguagem Python é **case sensitive** → Diferencia letras Maiúsculas de letras minúsculas
- A linguagem Python delimita os blocos por **indentação**:

Exemplo:

Este Bloco:

Tem este comando dentro dele

E este comando

...

Mas este comando não, pois está fora do bloco

Linguagem Python



- Assim como a língua portuguesa têm várias maneiras de expressar a mesma frase, algumas mais organizadas e mais simples de entender do que outras.

Exemplo: **cOnseGuimos enTeNder eSta FRaSE aPesar dE nÃO estAr BEm orGaNiZAda**

Como guia de **Boas Práticas**, a linguagem Python segue o PEP8:

<https://peps.python.org/pep-0008/>

Este guia orienta como escrever a linguagem de forma organizada e padronizada, apesar da linguagem funcionar mesmo se não seguir à risca o guia.

Princípios Orientadores (ZEN de Python)

Aqui estão alguns princípios orientadores para boas práticas na linguagem:

- Bonito é melhor que feio.
- Explícito é melhor que implícito.
- Simples é melhor que complexo.
- Complexo é melhor que complicado.
- Legibilidade conta.
- Casos especiais não são especiais o bastante para quebrar as regras.
- Diante da ambiguidade, recuse a tentação de adivinhar.
- Dever haver um — e preferencialmente apenas um — modo óbvio para fazer algo.
- Se a implementação é difícil de explicar, é uma má ideia
- Se a implementação é fácil de explicar, pode ser uma boa ideia

Referência: <https://wiki.python.org.br/TheZenOfPython>

IDE

- Para executar a linguagem python basta apenas um **Interpretador Python**, que irá ler na linguagem python e executar as tarefas descritas
- Para escrever a linguagem, pode ser feito em qualquer **Editor de Texto**, até mesmo no bloco de notas
- Porém, existem programas que unem um **Interpretador Python** junto com um **Editor de Texto** voltado a programação, estes softwares são chamados de **IDE** (Integrated Development Environment → Ambiente de Desenvolvimento Integrado)

Algumas IDEs de Python:

Vamos utilizar esta



Primeiro programa em Python



Após realizar a instalação da IDE, crie um arquivo com o Nome **Aula_1.py** e digite dentro do arquivo o seguinte texto: ***print("Ola Mundo!")***

Em seguida execute o programa:

Algoritmo

Executar

Pasta de Trabalho

Saída do código

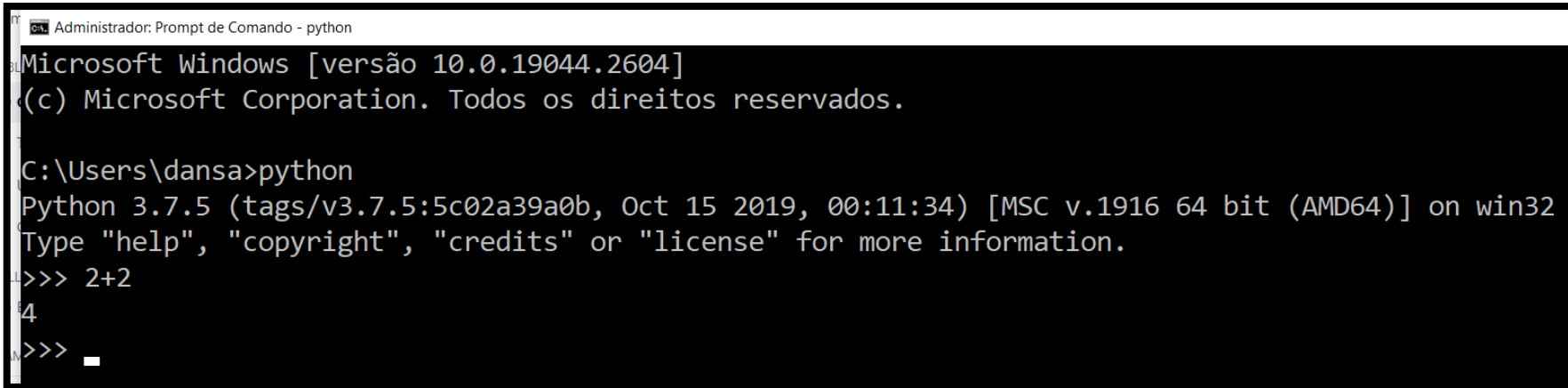


Interactive Mode

Outra forma de executar programas em python é utilizando o modo **Interactive Mode**, neste modo o programa é lido e executado a cada linha

Para fazer isso basta digitar no **prompt de comando**: **python**

Para testar, vamos digitar: **2+2** e então teclar **ENTER**



```
Administrador: Prompt de Comando - python
Microsoft Windows [versão 10.0.19044.2604]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\dansa>python
Python 3.7.5 (tags/v3.7.5:5c02a39a0b, Oct 15 2019, 00:11:34) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+2
4
>>> █
```

Interactive Mode

Se digitar algo errado, o compilador irá acusar o erro para auxiliar a entender o que houve de errado

```
>>> 5+  
      File "<stdin>", line 1  
        5+  
         ^  
SyntaxError: invalid syntax  
>>>
```

Para sair deste modo, basta digitar: **exit()** e teclar **ENTER**

```
>>> exit()  
  
C:\Users\dansa>
```

Tipos de dados

Existem diversos tipos de dados aplicáveis dentro de um programa, abaixo segue alguns deles:

- Integers (Inteiro)
-2, -1, 0, 1, 2, 3
- Float-point numbers (Número Racional)
-2.5, -1.2, 0.12, 1.3, 22.7, 12.0
- Strings (Texto)
'Ola Mundo', 'Daniel', 'a', 'abcdef', 'Jorge compra arroz no mercado', '57'
- Boolean (Verdadeiro | Falso)
True, False

Operadores matemáticos



Para executar contas matemáticas em Python, pode-se utilizar os operadores descritos na tabela abaixo:

| Operador | Operação | Exemplo | Resposta |
|----------|----------------------|-----------|----------|
| ** | Exponencial | $4 ** 3$ | 64 |
| % | Resto de Divisão | $22 \% 8$ | 6 |
| // | Quociente da Divisão | $22 // 8$ | 2 |
| / | Divisão | $22 / 8$ | 2.75 |
| * | Multiplicação | $3 * 5$ | 15 |
| - | Subtração | $8 - 3$ | 5 |
| + | Adição | $11 + 7$ | 18 |

Operadores matemáticos

A linguagem Python segue os mesmos princípios da matemática ao realizar as contas, então, por exemplo, a multiplicação ocorre antes da adição.

```
>>> 5*3+2  
17
```

A utilização de parênteses determina a ordem de resolução, assim como na matemática

```
>>> 5*(3+2)  
25
```


Operadores relacionais



Para executar comparações é necessário utilizar um operador relacional, que irá retornar **verdadeiro** ou **falso** dependendo da comparação executada

| Operador | Operação | Exemplo | Resposta |
|----------|------------------|----------|--------------|
| == | Igual a | 42 == 42 | True |
| != | Diferente de | 42 != 42 | False |
| < | Menor que | 22 < 8 | False |
| > | Maior que | 22 > 8 | True |
| <= | Menor ou igual a | 5 <= 5 | True |
| >= | Maior ou igual a | 8 >= 3 | True |

Operadores relacionais

É possível realizar comparações com contas matemáticas

```
>>> 5+3 == 15  
False
```

```
>>> 5*3 == 15  
True
```

Também é possível comparar tipos de dados diferentes

```
>>> '5' == 5  
False
```

```
>>> 5.0 == 5  
True
```

```
>>> 72 > 0052.2500  
True
```

```
>>> 'DANIEL' == 'Daniel'  
False
```

```
>>> 'Python' == 23  
False
```

Concatenação e Replicação

Concatenar é o nome dado a juntar, em programação é utilizado para juntar duas palavras ou textos lado a lado.

```
>>> 'Palmeiras ' + 'não tem' + ' mundial'  
'Palmeiras não tem mundial'
```

```
>>> 'meu nome é ' + 'Daniel'  
'meu nome é Daniel'
```

Replicar em programação é como uma multiplicação de texto.

```
>>> 'Joao'*5  
'JoaoJoaoJoaoJoaoJoao'
```

```
>>> '-' * 5 + 'PYTHON' + '-' * 5  
'-----PYTHON-----'
```

Criação de variáveis

Variáveis são **palavras-chave** que armazenam dados, existem diversos tipos de variáveis, condizentes aos tipos de dados existentes. Em python não é necessário dizer qual o tipo de variável ao criar ela, bastando apenas colocar um dado do tipo correto.

Usando o comando **type** é possível saber de qual tipo aquela variável é.

```
>>> ano = 2021
>>> type(ano)
<class 'int'>
```

```
>>> semana = 'sabado'
>>> type(semana)
<class 'str'>
```

```
>>> dinheiro = 12.5
>>> type(dinheiro)
<class 'float'>
```

Criação de variáveis

Uma vez que as variáveis armazenam valores, é possível comparar as variáveis, fazer contas matemáticas com elas, concatenar ou replicar.

```
>>> num1 = 5
>>> num2 = 3
>>> num1 + num2
8
>>> num1 * num2 + num1 ** 2
40
```

```
>>> nome = 'Joao'
>>> sobrenome = 'Cardoso'
>>> print('meu nome completo é ' + nome + ' ' + sobrenome)
meu nome completo é Joao Cardoso
```

```
>>> nome == 'Joao'
True
```


Criação de variáveis



Ao criar uma variável deve-se obedecer as seguintes regras:

- Só pode ser uma palavra (sem espaços)
- Só pode utilizar letras, números e sublinha (_)
- Não pode iniciar com um número

| Nomes válidos para variáveis | Nomes inválidos para variáveis |
|------------------------------|---|
| num_caracteres | num-caracteres (não pode hífen) |
| numCaracteres | num caracteres (não pode espaço) |
| conta3 | 3conta (não pode começar com número) |
| _42 | 42 (não pode começar com número) |
| SOMA_TOTAL | \$OMA_TOTAL (Caracteres especiais como \$ não são permitidos) |
| ola | 'ola' (Caracteres especiais como ' não são permitidos) |

Estilos de Programação

Por questão de organização e padronização, existem estilos predefinidos para criação de variáveis e funções, são eles:

- **camelCase**

Este estilo sempre começa com letras minúsculas e as próximas palavras tem a primeira letra maiúscula: **contaBancaria, nomeCompleto**.

- **PascalCase**

Este estilo sempre começa com a primeira letra Maiúscula em cada palavra: **ContaBancaria, NomeCompleto**.

- **snake_case**

Este estilo sempre utiliza letras minúsculas e separa as palavras com **sublinha** (_): **conta_bancaria, nome_completo**

- **kebab-case**

Este estilo sempre utiliza letras minúsculas e separa as palavras com **hífen** (-): **conta-bancaria, nome-completo**

Estilos de Programação

Em python, foram estabelecidas algumas regras para a criação de nomes de funções, variáveis, classes, etc.

- snake_case para **variáveis, funções e métodos**;
- PascalCase para **classes**;
- SCREAMING_SNAKE_CASE para constantes.

Mais informações: <https://dev.to/immurderer/guia-de-estilo-python-pep-8-lm8>

Funções

Funções são uma sequência de comandos que executam alguma tarefa e que tem um nome predefinido.

Por exemplo, a instrução `print('Joao')` é uma função que imprime a palavra Joao. A instrução `type('Joao')` é uma função que retorna o tipo de dado que contém dentro do parênteses.

Existem diversas funções dentro da base do python e outras que podem ser adicionadas através da inclusão de bibliotecas, também é possível criar suas próprias funções (veremos isso posteriormente).

Agora iremos conhecer algumas funções já existentes em python:

- **Função `input()`**
- **Função `len()`**
- **Função `str()`**
- **Função `int()`**
- **Função `float()`**

Função input()

Esta função armazena um conjunto de texto digitado no computador até receber o caracter `\n` (caracter que representa a tecla **ENTER**)

```
>>> nome = input()
Daniel
>>> print(nome)
Daniel
```

Este exemplo solicita ao usuário que digite seu nome, ao digitar e teclar **ENTER** o texto digitado será armazenado na variável **nome**.

Na linha seguinte, foi utilizado a função **print()** para exibir o valor armazenado na variável **nome**.

Função len()

Esta função exibe o valor de caracteres dentro de uma string.

Neste exemplo, utilizamos a função **print()** para exibir o número de caracteres dentro da variável **nome** utilizando a função **len()**.

```
>>> print(len(nome))  
6
```

Porém, como a função **len()** retorna um valor do tipo **int**, não podemos **concatenar** este valor diretamente pois dará erro.

```
>>> print('O nome ' + nome + ' tem ' + len(nome) + ' caracteres')  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: can only concatenate str (not "int") to str
```

Função `str()`

A função `str()` converte um tipo de variável para **string**

```
>>> num_caracteres = str(len(nome))
>>> type(len(nome))
<class 'int'>
>>> type(num_caracteres)
<class 'str'>
```

Desta forma, é possível realizar a instrução que havia dado erro anteriormente

```
>>> print('o nome ' + nome + ' tem ' + str(len(nome)) + ' caracteres')
o nome Daniel tem 6 caracteres
>>> print('o nome ' + nome + ' tem ' + num_caracteres + ' caracteres')
o nome Daniel tem 6 caracteres
```

Funções `int()` e `float()`

Assim como a função `str()`, as funções `int()` e `float()` servem para converter um tipo de dado em outro.

A função `int()` converte um tipo de variável para **integer (inteiro)**

```
>>> num_caracteres  
'6'  
>>> type(num_caracteres)  
<class 'str'>  
>>> int(num_caracteres)  
6  
>>> type(int(num_caracteres))  
<class 'int'>
```

A função `float()` converte outro tipo de variável para **float**

```
>>> float(42)  
42.0  
>>> float('5')  
5.0  
>>> float(num_caracteres)  
6.0
```