SEMANA PYTHONISTA

Seja disputado pelo mercado e comece ganhando 5x mais!

APOSTILA 2

★ DALTON PEIXOTO ★

APLICAÇÕES DO PYTHON

Inteligência Artificial

Big Data

Ciência de Dados

NETFLIX

Google

facebook

Automação

Nuvem

Segurança da Informação

ebay

aws

Dropbox

Web

Pesquisas Científicas

IoT
(Internet of Things)

YouTube

amazon alexa

Jogos

Mobile

Muito Mais

PRINCIPAIS BIBLIOTECAS E FRAMEWORKS

Inteligência Artificial

- Tensor Flow
- PyTorch
- Keras
- Scikit-learn

Big Data

- PySpark
- Kafka Python
- Pydoop

Ciência de Dados

- Pandas
- Matplotlib
- Scikit-learn
- SciPy

Automação

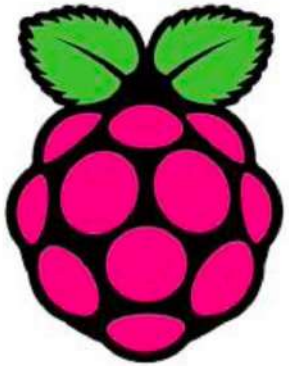
- Selenium
- Scrapy
- PyTest
- PyWinAuto
- AutoPy

Nuvem

- Boto3
- Google Cloud Python Client
- Azure SDK for Python
- s4cmd

Segurança da Informação

- Scapy
- Nmap
- Requests
- PaiMei



RaspberryPi

Web

- Django
- Flask
- CherryPy
- Bottle

Pesquisas Científicas

- Matplotlib
- Scipy
- Scikit-learn
- Numpy

IoT (Internet of Things)

- Raspberry Pi
- PySerial
- Alexa Skill Kit Sdk for Python
- Azure IoT Edge

Jogos

- PyGame
- PyOpenGL
- Pyglet
- Cocos2D (Python)

Mobile

- Kivy
- BeeWare
- Django-PWA

Muito Mais

Nessa aula você irá aprender:

- Os operadores de comparação no Python.
- O que são condicionais.
- Fluxo de execução de uma estrutura condicional.
- Laços de repetição do tipo **while** e **for**.
- Como criar e definir funções:
  - Sem parâmetros
  - Com parâmetros posicionais
  - Com parâmetros nomeados

Além disso também aprimoraremos os dois projetos da aula anterior deixando bem mais robustos:

- Calculadora de IMC
- Calculadora de média de notas de um aluno

E no final terão exercícios para praticar tudo o que aprendeu!

Operadores de comparação

- Os operadores de comparação são usados para comparar valores.
- Sempre retornam **True** ou **False**.

Operador	Conceito	Exemplo
>(Maior que)	Verifica se um valor é maior que outro	x > 5
<(Menor que)	Verifica se um valor é menor que outro	x < 5
== (Igual a)	Verifica se um valor é igual a outro	x == 5
!= (Diferente de)	Verifica se um valor é diferente de outro	x != 5
>= (Maior ou igual a)	Verifica se um valor é maior ou igual a outro	x >= 5
<= (Menor ou igual a)	Verifica se um valor é menor ou igual a outro	x <= 5

In [ ]:
```
4-1 <= 2    # False
3.5/3 != 1  # True
2**3 == 4*2 # True
4 / 2 != 2  # False
4 % 2 > 2   # False
```

Out[ ]: False

Os valores precisam ser do mesmo tipo para que a comparação retorne **True** Exemplo:

- Comparar **"1" == "1"** irá retornar **True** pois além dos valores serem iguais o tipo do dado é o mesmo nos dois lados da comparação, nesse caso ambos são strings.
- Mas comparar **"1" == 1** irá retornar **False** porque mesmo que o valor seja o mesmo o tipo da variável é diferente nos dois casos, do lado esquerdo temos uma string e do lado direito um número inteiro.

In [ ]:
```
"1" == "1"
```

Out[ ]: True

In [ ]:
```
"1" == 1
```

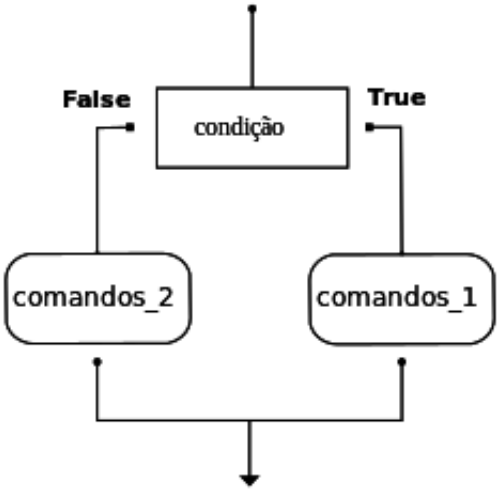
Out[ ]: False

Condicionais

Com a finalidade de escrever programas úteis, quase sempre temos a necessidade de verificar condições e alterar o comportamento do programa de acordo com os resultados das condições.

Comandos de seleção, algumas vezes também denominados de **comandos condicionais** nos dão essa habilidade.

A forma mais simples de seleção é o comando **if**. Ele é algumas vezes denominado de seleção binária uma vez que admite dois possíveis caminhos de execução.



In [ ]:
```
if True: # condição
    print('Só mostra quando verdadeiro') # comandos_1
else:
    print('Só mostra quando todas as opções anteriores são falsas')
# comandos_2
```

```
Só mostra quando verdadeiro
```

In [ ]:
```
idade = int(input('Qual a sua idade?'))
```

```
Qual a sua idade?25
```

In [ ]:
```
if idade < 12:
    print('crianca')
elif idade < 18:
    print('adolescente')
elif idade < 60:
    print('adulto')
else:
    print('idoso')
```

```
adulto
```

Melhorando o projeto da calculadora de IMC

CLASSIFICAÇÃO	IMC
Abaixo do Peso	Abaixo 18,5
Peso Normal	18,5 - 24,9
Sobrepeso	25 - 29,9
Obesidade Grau I	30 - 34,9
Obesidade Grau II	35 - 39,9
Obesidade Grau III ou Mórbida	Maior ou Igual 40

In [ ]:
```
peso = float(input('Qual o seu peso em kg?'))
```

```
Qual o seu peso em kg?72
```

In [ ]:
```
altura = float(input('Qual a sua altura em metros?'))
```

```
Qual a sua altura em metros?1.72
```

In [ ]:
```
imc = peso / altura**2
```

In [ ]:
```
if imc < 18.5:
    print('Você está abaixo do peso')
elif imc < 25:
    print('Parabéns! Você está com o peso normal')
elif imc < 30:
    print('Você está com sobrepeso')
elif imc < 35:
    print('Você está com obesidade de grau 1')
elif imc < 40:
    print('Você está com obesidade de grau 2')
else:
    print('Vocês está com obesidade de grau 3 também conhecida como
obesidade mórbida')
```

Laços de Repetição

1. While

O comando **while** faz com que um conjunto de instruções seja executado enquanto uma condição é atendida.

Quando o resultado dessa condição passa a ser falso, a execução do loop é interrompida.

In [ ]:
```
a = 0

while a < 3:
    print('o valor de a é:', a)
    a += 1

print('fim do while')
```

```
o valor de a é: 0
o valor de a é: 1
o valor de a é: 2
fim do while
```

Loopings Infinitos

Um loop ou laço infinito é aquele que apresenta **sempre uma condição de teste verdadeira**,ou seja, nunca termina!

Isto ocorre quando escrevemos acidentalmente uma condição que jamais será satisfeita, ou então esquecemos de alterar o valor da variável de controle do laço fazendo com que a condição permaneça **verdadeira para sempre**.

In [ ]:
```
# while True:
#    print('loop infinito')
```

Exemplo de loop inifinito em que esquecemos de alterar o valor da variável **a** no final do laço.

In [ ]:
```
# a = 0
# while a < 10:
#     print(a)
```

2. For

O laço **for** nos permite percorrer os itens de uma coleção e, para cada um deles, executar o bloco de código declarado no loop.

In [ ]:
```
lista = [1,2,3,'a', 'b', 'c']
```

In [ ]:
```
# Podemos usar o FOR para percorrer todos os elementos de uma lista
for elem in lista:
  print(elem)
```

```
1
2
3
a
b
c
```

In [ ]:
```
# Ou simplesmente criamos um RANGE() com a quantidade de vezes que queremos que o laço seja executado
for elem in range(10):
  print(elem)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [ ]:
```
# Podemos também definir o ponto de partida em RANGE()
for elem in range(7, 10):
  print(elem)
```

```
7
8
9
```

Melhorando o projeto da calculadora de média das notas de um aluno

In [ ]:
```
materias = ['português', 'matemática', 'inglês', 'história']
soma = 0

for materia in materias:
  nota = float(input(f'Qual a nota do aluno em {materia}'))
  soma += nota
```

```
Qual a nota do aluno em português9
Qual a nota do aluno em matemática2.8
Qual a nota do aluno em inglês7.5
Qual a nota do aluno em história5
```

In [ ]:
```
media = soma / len(materias)
```

In [ ]:
```
print('A média do aluno é', media)
```

```
A média do aluno é 6.075
```

Também poderíamos usar uma f-string para exibir o resultado da nossa calculadora de médias

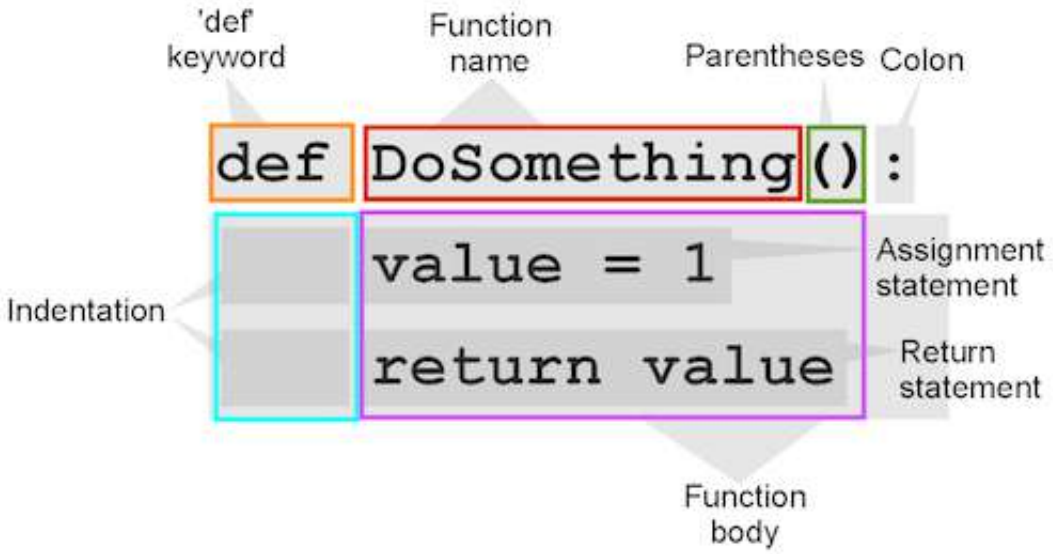
In [ ]:
```
print(f'A média do aluno é {media}')
```

```
A média do aluno é 6.075
```

Funções

Uma função é uma sequência de comandos que executa alguma tarefa e que tem um nome.

A sua principal finalidade é nos ajudar a organizar programas em pedaços que correspondam a como imaginamos uma solução do problema.



In [ ]:
```
# A funcao nao é executada
def foo():
    valor = 1
    print(valor)
```

In [ ]:
```
# executa quando chamamos a funcao
foo()
```

```
1
```

In [ ]:
```
# funcoes com argumentos obrigatorios
def anfitriao(nome_convidado):
    print(f'Olá, {nome_convidado} seja muito bem vindo ao curso de Python')
```

In [ ]:
```
# Chama a funcao
anfitriao('Dalton')
anfitriao('Programador Aventureiro')
```

```
Olá, Dalton seja muito bem vindo ao curso de Python
Olá, Programador Aventureiro seja muito bem vindo ao curso de Pyth
on
```

Anfitrião foi definido com um argumento obrigatório, sendo assim receberemos um erro se tertarmos executar a função sem passar esse argumento obrigatório

In [ ]:
```
anfitriao()
```

```
---------------------------------------------------------------------
---------
TypeError                                 Traceback (most recent c
all last)
<ipython-input-36-d8ae6378bb45> in <module>()
----> 1 anfitriao()

TypeError: anfitriao() missing 1 required positional argument: 'no
me_convidado'
```

In [ ]:
```
# funcoes com argumento opcional
def anfitriao2(nome_convidado='Fulano'):
    print(f'Olá, {nome_convidado} seja muito bem vindo ao curso de
Python')
```

In [ ]:
```
anfitriao2('Dalton')
```

```
Olá, Dalton seja muito bem vindo ao curso de Python
```

In [ ]:
```
anfitriao2()
```

```
Olá, Fulano seja muito bem vindo ao curso de Python
```

Uma função pode ter diversos parâmetros. Para passar mais de um parâmetro para uma função basta separar cada um deles usando uma vírgula

In [ ]:
```
def func(parametro1, parametro2, parametro3, parametro4):
    pass
```

Exercícios

1. Faça um programa que calcule a soma entre todos os números que são múltiplos de três e que se encontram no intervalo de 1 até 500.
2. Crie um programa que leia o ano de nascimento de sete pessoas. No final, mostre quantas pessoas ainda não atingiram a maioridade e quantas já são maiores.
3. Supondo que a população de um **país A** seja da ordem de **80.000** habitantes com uma taxa anual de crescimento de **3%** e que a população de **B** seja **200.000** habitantes com uma taxa de crescimento de **1.5%**. Faça uma função que calcule e escreva o número de anos necessários para que a população do **país A** ultrapasse ou iguale a população do **país B**, mantidas as taxas de crescimento. Ela deve receber como parâmetros obrigatórios a quantidade de habitantes do país A e B, além das taxas de crescimento de cada país.
4. Faça uma função que informe a quantidade de dígitos de um determinado número inteiro informado.

SIGA-NOS NAS REDES SOCIAIS

Clique no ícone para ser redirecionado:

YouTube

Aqui eu compartilho vídeos com muito conteúdo para ajudar você a conquistar seus objetivos o mais rápido possível.

Blog

Este é o portal de conteúdo, você vai encontrar diversos artigos passo a passo com estratégias para seu desenvolvimento pessoal na programação sobre diversas linguagens.

Instagram

Aqui eu compartilho imagens dos meus bastidores, do meu dia a dia, pequenos vídeos com dicas etc.

Facebook

Aqui você verá sempre a atualização das postagens do blog, comentários e engajamento dos meus alunos, fotos e dicas rápidas de programação.

</PROGRAMADOR AVENTUREIRO>