Introdução ao Python

Repositório : https://github.com/Sampa-USP/Intro-Python.git

Dúvidas : https://github.com/Sampa-USP/Intro-Python/issues

Sumário de hoje:

- tipos primitivos (int, float, str e bool);
- sintaxe básica (soma, divisão, multiplicação e exponenciação);
- estrutura condicional e laços (if, while e for);
- estrutura de dados (dicionários e listas);
- funções (operações e escopo);

"E no principio era trevas, no inicio do inicio..."

- Python : utilizaremos a partir do 3.x;
- notebook vs python :
  - python :
    - arquivo termina em .py;
    - bom para pós processamento e execução de códigos 'pesados';
    - executado por inteiro;

```
python arquivo.py
```

  - notebook :
    - arquivo termina em .ipynb;
    - executado em 'células';
    - bacana para códigos que você quer conhecer seus dados

```
jupyter notebook
```

"E no principio era trevas, no inicio do inicio..."

- use o conda para evitar conflitos instalar conda;
- **JAMAIS** use o python do seu SO, instale pacotes com pip ou conda:
  - PIP Python Package Index :

```
pip install package_name
```

  - CONDA

```
conda install package_name
```

Tipos primitivos:

- Basicamente, número e letra.

```
2 #int
2.0 #float
True #bool
'paulo' #str
```

- Para atribuir, basta

```
x = 2
y = 2.0
z = True
k = 'paulo'
```

- OBS : variável só deve começar com letras ou "_":

```
2x = 3 #ERRO!!!
@a = 3 #ERRO!!!
)a = #ERRO!!!
```

Sintaxe Básica:

- Uma vez definida uma variável, o que fazer com ela ? Operações!

```
x = 5 # x = 5
x_dot_2= x*2 # x_dot_2 = 10
x_dot_x = x*x # x_dot_x = 25
x_square = x**2 # x_square = 25

nome = 'paulo' # nome = paulo
dois_nome = 'paulo' * 2 # dois_nome = paulopaulo
soma_nome = 'andreia' + 'paulo' # soma_nome = andreiapaulo

_nome = nome + x #erro
_nome = nome + str(x) # _nome = paulo5
_nome = nome * str(x) # erro
_nome = nome * x # _nome = paulopaulopaulopaulopaulo
```

Exercício 1 :

- dadas duas variáveis, troque seus valores; **note1.ipynb**
- soma de tipos booleanos; **note1.ipynb**
- operadores em python ; **note1.ipynb**
- resultado de operadores aritméticos comparativos ; **note1.ipynb**

Estrutura Condicional :

- Python possui três operadores de estrutura condicional **if**, **elif** e **else**. Toda estrutura condicional deve necessáriamente começar com **if**, mas é livre para terminar com qualquer um. Por exemplo:

```
x = 2
if x>0: #if possui condião explícita
    print("Valor é positivo")
elif x==0: #elif possui condição explícita
    print("Valor é zero")
else : #não possui confição
    print("Valor negativo")
```

Estrutura de Dados :

- Um dicionário e uma lista em python guardam quaisquer tipos de dados. Podemos possuir uma lista de funções, classes, números, letras, letras e números, classes e funções e etc... para acessar esses itens você depende da posição deles na lista a começar do **0**.
- O dicionário também, mas para acessar você pode utilizar uma 'chave'.

```
x = 2
lista = [[1,2,3],x] #definida entre barras
print(lista[0]) # [1,2,3]
print(lista[0][2]) # [3]
print(lista[1]) # x
print(lista[2]) # erro!!!
```

Estrutura de Dados :

```
x = 2
dicionario = {"x" : x, "lista" : [1,2,3]} #definido entre colchetes
print(dicionario['x']) # 2
print(dicionario['lista']) # [1,2,3]
print(dicionario['lista'][1]) # 2
```

Definindo Funções :

- Vai precisar de algo várias vezes e não quer ficar repitindo código ? Parabéns, você precisa de uma função! Funções são objetos simples de altíssima abstração em python. Podem ou não retornar um valor.

```
def nome_da_funcao(argumento1, argumento2, ...): #funções são definidas com a palavra mágica def
    # bloco de código da função
    return valor_de_saida_1,valor_de_saida_2,... # return é tudo que sua função precisa retornar. Pode ser QUALQUER tipo de dado.
```

Escopo de variáveis :

- O escopo de uma variável pode ser global ou local. Global se definida no código e local se definida em uma função, por exemplo :

```
def func1():
    x = 4
    print(x)

def func2():
    print(x)

x = 2
print(x)
func1()
func2()
print(x)
```

O retorno será:

```
2
4
2
2
```

Exercício 2 :

- escreva uma função que encontre o menor valor de uma lista; **note2.ipynb**
- uma função para calcular o n-ésimo número da sequência de fibonacci; **note2.ipynb**
- defina uma função que a partir de duas matrizes, retorne o produto matricial; **note2.ipynb**