Workshop_Python

August 5, 2018

1 Workshop Linux-Python

2 Objetivos

2.0.1 - Ensinar conceitos básicos sobre Python, sua sintaxe e estrutura

2.0.2 - Ensinar conceitos básicos sobre trabalhos com matrizes e dataframes

2.0.3 - Conceitos básicos, e mais avançados, sobre visualização de dados

3 Por que Python?

3.0.1 - Fácil de aprender

3.0.2 - É bem documentado e gratis.

3.0.3 - Popular (é fácil de encontrar ajuda na internet), grande comunidade.

3.0.4 - Rápido !?

3.0.5 - Scripts são portáveis (podem ser rodados em qualquer SO).

3.0.6 - Uma grande quantidade de bibliotecas e aplicações.

4 Sintaxe básica

4.0.1 - Python é uma linguagem de "Alto nível", isto é, ela é o mais próximo possível da linguagem humana.

4.0.2 - É possível rodar comandos interativamente ou em scripts.

```
In [1]: # Primeiro print no Jupyter

        print("Olá, Python!")
```

```
Olá, Python!
```

Variáveis

4.0.3 - Uma varíavel é algo que pode ser mudado!!!

4.0.4 - É uma forma de remeter a locação de memória por um programa de computador.

4.0.5 - Ela armazena valores, possui um nome (identificador) e um tipo de dado.

4.0.6 - Enquanto o programa estiver rodando, a variável pode ser acessada e, as vezes, mudada.

4.0.7 - Python não é uma linguágem "strongly-typed", isso significa que o tipo de dados armazenados em variaveis pode ser mudado.

5 Variáveis e idenfiticadores

5.0.1 - Algumas pessoas confundem variáveis e identificadores.

5.0.2 - Mas identificadores são nomes de variáveis, variáveis possuem outros atributos.

5.0.3 - Identificadores também são utilizados por funções, módulos, bibliotecas, etc...

5.0.4 - Um identificador válido é uma sequencia de caracteres de qualquer tamanho com:

5.0.5 a) Um caractere inicial(um underscore ou letras em maíusculo ou minúsculo).

5.0.6 b) Letras seguintes podem ser qualquer tipo de caractere.

5.0.7 c) Palavras chaves de Python não podem ser utilizadas como identificadores! Ex: and, as, assert, break, class, continue, def, del, elif, else, except, for, if, in, is, lambda, not, or, pass, return, try, with.

```
In [2]: # Exemplos de variáveis

        i = 30
        J = 32.1

        uma_string= "string"
```

6 Tipos de dados básicos em Python

6.1 Numéricos:

6.1.1 1. int: integers, ex: 610, 9580

6.1.2 2. Pontos flutuantes, ex: 42.11, 2.5415e-12

6.1.3 3. Complexos, ex: x = 2 + 4i

6.2 Sequências:

6.2.1 1. str: Strings (sequencia de caracteres), ex: "ABCD", "Olá mundo!", "C_x-aer"

6.2.2 2. list

6.2.3 3. tuple

6.3 Booleana:

6.3.1 True or False

6.4 Mapping:

6.4.1 dict (dictionary)

7 Números

7.0.1 int (integers): números positivos ou negativos sem ponto decimal.

```
In [3]: a_number = 35
        type(a_number)
```

```
Out[3]: int
```

7.0.2 - Note que se adicionarmos " " o tipo do dado vai mudar, ele não será um integer mas sim uma string.

```
In [4]: a_number = "35"
        type(a_number)
```

```
Out[4]: str
```

7.0.3 - float (valores de pontos flutuantes): números reais com casas decimais (ex: 23.567) também podem ser notações científicas (1.54e2 – é o mesmo que 1.54 x 100 e o mesmo que 154)

```
In [5]: um_float=23.567
        type(um_float)
```

```
Out[5]: float
```

```
In [6]: outro_float = 1.54e20
        type(outro_float)

Out[6]: float

In [7]: outro_float

Out[7]: 1.54e+20
```

8 Conversões de números

```
In [8]: um_float

Out[8]: 23.567

In [9]: int(um_float)

Out[9]: 23

In [10]: um_float

Out[10]: 23.567

In [11]: # modificações devem ser salvas em uma nova variável

         um_int = int(um_float)

In [12]: um_int

Out[12]: 23

In [13]: str(um_float)

Out[13]: '23.567'
```

9 Prática:

- Crie uma variável numérica do tipo float
- A converta para tipo int

10 Operações numéricas

10.0.1 - Os operadores +,*,-,/ são aritméticos para variáveis numéricas.

```
In [14]: x=55
         y=30
```

```
In [15]: x+y

Out[15]: 85

In [16]: x-y

Out[16]: 25

In [17]: x*y

Out[17]: 1650

In [18]: x/y  # por que 1 ??

Out[18]: 1.8333333333333333

In [19]: float(x)/float(y)

Out[19]: 1.8333333333333333
```

10.0.2 - Alguns operadores extras:

10.0.3 % (Módulo) retorna o resto de uma divisão.

10.0.4 ** (Expoente) retorna o resultado de um cálculo exponencial.

```
In [20]: x%y

Out[20]: 25

In [21]: x**y

Out[21]: 162510222465604611845303361805165186524391174316406 25
```

11 Prática

- Crie 3 variáveis com os valores 4, 12.4 e 30.
- Calcule a soma de todos e armazene o resultado em uma nova variável.
- Multiplique o resultado por 5 e armazene o resultado em uma nova variável.

12 Strings

12.0.1 - Strings são marcadas por aspas.

12.0.2 - Envolta por aspas simples (’):

```
In [22]: print('Esta é uma string com aspas simples')

Esta é uma string com aspas simples
```

12.0.3 - Envolta por aspas duplas (''):

```
In [23]: print("This is a string with double quotes")
```

```
This is a string with double quotes
```

12.0.4 - Envolta por 3 caracteres, usando aspas simples ou duplas:

```
In [24]: print('''Uma string com aspas triplas pode se extender por multiplas linhas, podendo co
```

```
Uma string com aspas triplas pode se extender por multiplas linhas, podendo conter também aspas
```

12.0.5 - Uma string em Python consiste em uma série ou sequencia de caracteres - letras, números e caracteres especiais.

12.0.6 - Strings podem ser indexadas, o primeiro caractere de uma string tem o index 0.

```
In [25]: str_1= "Uma string"
```

```
In [26]: str_1
```

```
Out[26]: 'Uma string'
```

```
In [27]: str_1[0]
```

```
Out[27]: 'U'
```

```
In [28]: str_1[3]
```

```
Out[28]: ' '
```

```
In [29]: len(str_1)
```

```
Out[29]: 10
```

12.0.7 - Último caractere:

```
In [30]: str_1[-1]
```

```
Out[30]: 'g'
```

12.0.8 - Isso só foi possível porque os indices também podem ser lidos a partir da direita ao usar valores negativos:

12.0.9 "STRING"

[-6],[-5],[-4],[-3],[-2],[-1]

12.0.10 Diferente do modo normal, a partir da esquerda:

12.0.11 "STRING"

[0],[1],[2],[3],[4],[5]

13 Strings são "imutáveis"

13.0.1 - Assim como em Java, strings em Python não podem ser modificadas.

```
In [31]: s = "Algumas coisas são imutáveis!"
         s[-1] = "."
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-31-0f4fd2e1620b> in <module>()
  1 s = "Algumas coisas são imutáveis!"
----> 2 s[-1] = "."


TypeError: 'str' object does not support item assignment
```

14 Operações com strings

14.0.1 - Concatenação: ao usarmos o operador numérico + é possível concatenar 2 ou mais strings:

14.0.2 Atenção para a falta de espaço

```
In [ ]: "Hello" + "World"  #  <- *Attention to the absence of space*
```

14.0.3 - Repetição- É possivel repetir uma string n vezes ao usar o operador *:

```
In [ ]: "OláMundo" * 3
```