Workshop Introdução ao Python

Code made simple!

Orador:
Ricardo Tonet, MIEB
Organizado por:

Conteúdo

- Apresentação Python
- Ambiente de Dev e bibliotecas
- Sintaxe:
  - Variáveis e tipos de dados
  - Operadores
  - Input/Output
  - Estruturas de controlo
  - Estruturas de dados
  - Funções
  - Ficheiros
- Projecto

Python: a linguagem

- Criada em 1991 por Guido Van Rossum
- Open Source
- Linguagem interpretada
- General-purpose
- Vários paradigmas: oop, imperativo, funcional e procedimental
- Dynamic typing

Python: filosofia

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

Python: onde é usada

- Web Development: Django, Flask, ...
- Game Development: Pygame, ...
- Data Analytics & Machine Learning
- Science & Engineering
- Scripting
- Software development: Gimp, FreeCad, ...
- More...

Python: Ambiente Dev.

IP[y]:

Python: Standard library

- Biblioteca principal de Python que contém módulos standard para além da linguagem.

- Alguns módulos importantes:
  - string
  - datetime
  - array
  - math
  - random
  - os.path
  - time
  - e muita outras...

Python: Standard library

- Para além dos módulos, ainda existem funcões standard que estão sempre disponíveis:
  - open()
  - len()
  - input()
  - print()
  - range()
  - round()
  - min()
  - str()
  - e outras...

Python: “Hello World!”

```
print("Hello World!")
```

Tipos de dados

```
Booleanos: True, False
Inteiros: 10, 2342, -7842, 23423534534L
Reais: 3.14e-10, .001, 10., 1E3
Complexos: 2J, 1+5j, 23+.2j
Octal: 0177, 01234522222L
Hexadecimal: 0xFF, 0x23AA
Binary: 0b0011101, 0b101
Strings: 'python', "Isto é fácil!",
          """Multiline string"""


Saber mais em: https://docs.python.org/3/library/stdtypes.html
```

Strings

```
>>> "hello"+"world" "helloworld" # concatenation
>>> "hello"*3 "hellohellohello" # repetition
>>> "hello"[0] "h" # index
>>> "hello"[-1] "o" # starting from the end
>>> "hello"[1:4] "ell" # slicing
>>> len("hello") 5 # size
>>> "hello" < "jello" True # comparison
>>> "e" in "hello" True # searching
>>> "special characters: \n \t etc"
```

Saber mais em: https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

Variáveis

```
# <variavel> = <valor>
# Sem inicialização ou declaração prévia
>>> duzia = 12
# Nomes válidos
>>> du_zia, duzia12, meiaDuzia, ...
# Nomes inválidos
>>> 12_duzia, meia duzia
# Não têm tipo definido
>>> a = 1, a = [], a = 'hello'
>>> a = True
```

Saber mais em: https://docs.python.org/3/reference/lexical_analysis.html#identifiers

Operadores

Aritméticos:

+, -, *, /, //, %, **

Comparação:

<, >, <=, >=, !=, ==

Lógicos:

and, or, not

Atribuição:

=, +=, -=, *=, /=, //=, %=, **=, ...

Binários:

<<, >>, &, |, ^, ~

Saber mais em: https://docs.python.org/3/reference/lexical_analysis.html#operators

Output

```
# print('<texto>') ou print("<texto>")
>>> print('Estamos a aprender python!')
>>> print("I'm the teacher today! Yeay!")
>>> print('A palavra "casa" tem 4 letras.')
>>> print("Diz-se \"louça\" ou \"loiça\"?")
>>> print('I can\'t learn this!')
```

Saber mais em: https://docs.python.org/3/library/functions.html#print

Formatted Output

```
# '<texto> {} <texto>'.format(<valor>)
>>> 'O workshop demora {} horas'.format(3)
Out: 'O workshop demora 3 horas'
>>> 'Hora: {}:{}:{}'.format(hour, min, sec)
Out: 'Hora: 14:35:24'
>>> '{:.5}'.format('xylophone')
Out: xylop
```

Saber mais em: https://pyformat.info/

Input

```
# nome_var = input('<texto>')
>>> nome = input('Insira o seu nome: ')
Out: Insira o seu nome: ricardo
>>> print(nome)
Out: 'ricardo'
```

Saber mais em: https://docs.python.org/3/library/functions.html#input

Comentários

```
>>> # isto é um comentario
>>> x = 5 # x representa a var. independente
```

Exercícios

1) Obter o 2º, 3º e 4º caracteres da string "Anatomia".

2) Concatenar as strings 'ester', 'no', 'cleido', 'mas', 'toideu' e indicar o tamanho da string final.

3) Comparar as palavras 'casa' e 'Casa', com <, > e =.

4) Ler uma string da consola e apresentar o seu tamanho formatado na string: "A minha string tem xx letras".

Exercícios

5) Verificar se a palavra biomédica está presente no seguinte texto : "A Engenharia biomédica é uma área que integra princípios das ciências exatas e ciências da saúde."

6) Ler altura e peso da consola e calcular IMC com o valor formatado. IMC = altura / massa² (m/kg²).

7) Listar o valor lógico da inclusão do IMC nas categorias seguintes:

- [17; 18,5[- Magreza leve: True or False?
- [18,5; 25[- Saudável: True or False?
- [25; 30[- Sobrepeso: True or False?

Exercícios: Solução 1

```
# Solução ex.1
str = 'Anatomia'
print(str[1])
print(str[2])
print(str[3])
```

Exercícios: Solução 2

```
# Solução ex.2
print(len('ester'+'no'+'cleido'+'mas'+'toideu'))
```

Exercícios: Solução 3

```
# Solução ex.3
print('casa' < 'Casa')
print('casa' > 'Casa')
print('casa' == 'Casa')
```

Exercícios: Solução 4

```
# Solução ex.4
str = input('Insira uma string: ')
print('A minha string tem {}
letras'.format(len(str)))
```

Exercícios: Solução 5

```
# Solução ex.5
str = "A Engenharia biomédica é uma área que
integra princípios das ciências exatas e
ciências da saúde."
print('biomédica' in str)
```

Exercícios: Solução 6

```
# Solução ex.6
altura = float(input('Insira a altura: '))
massa = float(input('Insira a massa: '))
imc = altura / massa ** 2
print(imc)
```

Exercícios: Solução 7

```
# Solução ex.7
print('Magreza leve: '+str(imc >= 17 and imc <
18.5))
print('Saudável: '+str(imc >= 18.5 and imc <
25))
print('Sobrepeso: '+str(imc >= 25 and imc < 30))
```

Estruturas controlo: if/elif/else

```
# if <condicao>: <bloco> [elif: <bloco>] [else: <bloco>]
if imc > 25:
   print('Está muito pesado!')
elif imc < 18.5:
   print('Tem que comer mais!')
else:
   print('Está no ponto!)
```

Saber mais em: https://docs.python.org/3/reference/compound_stmts.html#the-if-statement

Estruturas controlo: for

```
# for <var> in <iter>: <bloco> [break][continue]
for i in range(10):
   print(i) # prints numbers from 0 to 9

for i in range(1,11):
   for j in range(1,11):
      print('{}x{}={}'.format(i,j,i*j))
   print("\n")
```

Saber mais em: https://docs.python.org/3/reference/compound_stmts.html#the-for-statement

Estruturas controlo: while

```
# while <condicao>: <bloco> [break][continue]
i = 1
while i <= 10:
    print(i)
    i += 1
```

Saber mais em: https://docs.python.org/3/reference/compound_stmts.html#the-while-statement

Estruturas controlo: do...while

```
# não existe em python
# emulacao:
i = 1
while True:
    print(i)
    i += 1
    if i > 10:
        break
```

Estruturas controlo: switch

```
# não existe em python
# emulacao:
num = 2
opcoes = {
   1: 'um',
   2: 'dois',
   3: 'tres'
}
print opcoes.get(num, 'Numero desconhecido')
```

Estruturas controlo: switch

```
# não existe em python
# emulacao:
func = 'func1'
opcoes = {
   'func1': calculate_imc,
   'func2': get_blood_pressure,
   'func3': save_to_file
}
f = opcoes.get(func, lambda 'funcao errada')
print f()
```

Estruturas dados: listas

```
>>> a = [0,1,2,3,4,5,6,7]
>>> a[3] # 3
>>> a[-1] # 7
>>> a[2:4] # [2, 3]
>>> a[1:] # [1, 2, 3, 4, 5, 6, 7]
>>> a[:3] # [0, 1, 2]
>>> a[:] # [0,1,2,3,4,5,6,7] # copies the
sequence.
>>> a[::2] # [0, 2, 4, 6] # even.
>>> a[::-1] # [7,6,5,4,3,2,1,0] # reverse order.
```

Estruturas dados: listas

```
>>> a = list(range(5)) # [0,1,2,3,4]
>>> a.append(5) # [0,1,2,3,4,5]
>>> a.pop() # [0,1,2,3,4]
Out: 5
>>> a.insert(0, 5.5) # [5.5,0,1,2,3,4]
>>> a.pop(0) # [0,1,2,3,4]
Out: 5.5
>>> a.reverse() # [4,3,2,1,0]
>>> a.sort() # [0,1,2,3,4]
```

Saber mais em: https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

Estrut. dados: list comprehensions

```
# var = [<transform> <iteration> <filter>]
# Normal
impares = []
for i in range(10):
    if i % 2 == 0:
    impares.append(i)
# List comprehension
impares = [i for i in range(10) if i % 2 == 0]

# criar vector de nºs quadrados
quad = [i**2 for i in range(10)]
Out: [0 1 4 9 16 25 36 49 64 81]
```

Saber mais em: https://docs.python.org/3/tutorial/datastructures.html (ponto 5.1.3)

Estrut. dados: list comprehensions

```
# A sintaxe permite vários transforms,
iterations e filters.
prod = [x*y for x in [1,2,3] for y in range(10)]
Out: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 2, 4, 6,
8, 10, 12, 14, 16, 18, 0, 3, 6, 9, 12, 15, 18,
21, 24, 27]

filter = [l for l in 'Hello World' if l != 'l'
if l > 'd']
Out: ['e', 'o', 'o', 'r']
```

Saber mais em: https://docs.python.org/3/tutorial/datastructures.html (ponto 5.1.3)

Estruturas dados: tuples

```
# Funcionam exactamente como listas
# São imutáveis
# Podem ser usados como chave em dicionários
>>> a = (0,1,2,3,4,5,6,7)
>>> a = tuple(range(5)) # (0,1,2,3,4)
```

Saber mais em: https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

Estruturas dados: dicionários

```
# Hash tables, "associative arrays"
>>> d = {"name": "Ana", "age": "11"}
>>> d["name"] # Ana
>>> d["number"] # generates a KeyError exception
# Delete, insert, overwrite :
>>> del d["age"] # {"name": "Ana"}
>>> d["grade"]="5" # {"name": "Ana", "grade":
"5"}
>>> d["name"]="Diana" # {"name": "Diana",
"grade": "5"}
```

Saber mais em: https://docs.python.org/3/library/stdtypes.html#mapping-types-dict

Estruturas dados: dicionários

```
# Chaves, valores, itens:
>>> d.keys() # ["name", "grade"]
>>> d.values() # ["Diana", "5"]
>>> d.items() # [("name","Diana"),("grade","5")]
# Verificar existência de chave:
>>>"name" in d # True
>>>"spam" in d # False
# Valor pode ser qq coisa. Chave tem q ser
imutável (!)
>>> {"name":"Alice", "age":15,
("hello","world"):1, 42:"yes", "flag":
["red","white","blue"]}
```

Estruturas dados: sets

```
>>> s1 = set([4, 6, 9])
>>> s2 = set([1, 6, 8])
>>> s1.union(s2) # set([1, 4, 6, 8, 9])
>>> s1 | s2 # set([1, 4, 6, 8, 9])
>>> s1.intersection(s2) # set([6])
>>> s1 & s2 # set([6])
```

Saber mais em: https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset

Exercícios

1) Ler IMC da consola e escrever uma mensagem consoante o intervalo do IMC.

2) Ler lista de valores de valores de IMC e achar o max e min:

lista de IMCs: [12, 17.5, 22, 26, 15.4]

3) Ordenar a lista anterior por ordem crescente

4) Criar dicionario com nome paciente, altura e peso, e calcular o IMC desse paciente.

5) Repetir o exercicio anterior mas usando tuples.

6) Criar uma lista de números naturais ímpares até 50.

Exercícios

7) Multiplicar cada elemento da lista [1, 2, 3] por 3 e colocar numa nova variável.

8) Criar uma lista da primeira letra de cada palavra do vector: words = ["this","is","a","list","of","words"]

9) Verificar se o n.º 10 existe no dicionário e indicar a sua chave:d = {'a':1, 'b':11, 'c':53, 'd':10, 'e':7}

10) Converter todos os caracteres da lista para minúsculas.
l = ['M', 'I', 'N', 'U', 'S', 'C', 'U', 'L', 'A', 'S']

Exercícios: Solução 1

```
imc = input('O seu imc: ')
if imc >= 17 and imc < 18.5:
   print('Magreza leve')
elif imc >= 18.5 and imc < 25:
   print('Saudável')
elif imc >= 25 and imc < 30:
   print('Sobrepeso')
else:
   print('Fora da escala!!')
```

Exercícios: Solução 2

```
imcs = [12, 17.5, 22, 26, 15.4]
print('Min: '+str(min(imcs)))
print('Max: '+str(max(imcs)))
```

Exercícios: Solução 3

```
imcs = [12, 17.5, 22, 26, 15.4]
sorted_imcs = sorted(imcs)
print(sorted_imcs)

# Outra opcao
imcs = [12, 17.5, 22, 26, 15.4]
imcs.sort()
print(imcs)
```

Exercícios: Solução 4

```
paciente = {
   'nome': 'ricardo',
   'peso': 80, # Kg
   'altura': 1.80 # metros
}
imc = paciente['altura'] / paciente['peso']**2
print(imc)
```

Exercícios: Solução 5

```
paciente = ('ricardo', 80, 1.80)
imc = paciente[2] / paciente[1]**2
print(imc)
```

Exercícios: Solução 6

```
nums = []
for i in range(1,51):
   if i % 2 != 0:
      nums.append(i)
print(nums)

# outra forma
nums = [i for i in range(1,51) if i%2!=0]
print(nums)
```

Exercícios: Solução 7

```
list = [1,2,3]
multiplied = [item*3 for item in list]
print(multiplied)
```

Exercícios: Solução 8

```
words = ["this","is","a","list","of","words"]
items = [ word[0] for word in words ]
print(items)
```

Exercícios: Solução 9

```
d = {'a':1, 'b':11, 'c':53, 'd':10, 'e':7}
print(10 in d.values())
keys = [k for (k, v) in d.items() if v == 10]
```

Exercícios: Solução 10

```
l = ['M', 'I', 'N', 'U', 'S', 'C', 'U', 'L',
'A', 'S']
ll = [c.lower() for c in l]
```

Funções

```
# def <nome_funcao> (arg1, arg2, ...): <bloco>
[return <valor>]
def soma(v1, v2):
   return v1+v2
>>> soma(2,3) # 5

def cumsum(list):
   val = 0
   for i in range(len(list)):
      val += list[i]
   return val
```

Módulos

```
# import <nome_modulo>
# from <nome_modulo> import <func1>,<func2>,...

# Importar módulo completo
>>> import string
>>> from string import *
>>> string.join(array_caracteres)
# Importar apenas algumas funções do módulos
>>> from string import join
>>> join(array_caracteres)
```

Saber mais em: https://docs.python.org/3/reference/import.html

Exercícios

1) Criar função para calcular o IMC.

2) Criar função para calcular calorias semanais com base na idade. Cal = [(20, 1200), (25, 1800), (30, 1600), (40, 1300)]

3) Criar função para verificar existência de diabetes com base nos níveis de glicose no sangue

Exercícios: Solução 1

```
def imc(height, weight):
    return weight / height ** 2
```

Exercícios: Solução 2

```
def weekly_cal(age):
   cal = [(20, 1200), (25, 1800), (30, 1600),
(40, 1300)]
   age_cal = [calv for (agev,calv) in cal if agev
== age][0]
   return age_cal*7
```

Exercícios: Solução 3

```
def check_diabetes(patient):
    glucose = float(patient['glucose'])

    if glucose < 70:
        print('-> Está com hipoglicemia.')
    elif glucose >= 70 and glucose < 100:
        print('-> Está normal.')
    elif glucose >= 100 and glucose < 126:
        print('-> Está com pré-diabetes!')
    else:
        print('-> Está com diabetes!!!')
```

Ficheiros: Leitura

```
# f = open('<nome_ficheiro>', 'modo_abertura')
# modo_abertura = {'r', 'w', 'a'}

f = open('dados.txt', 'r')
for line in f.readlines():
   print(line)
f.close()
```

Ficheiros: Escrita

```
# f = open('<nome_ficheiro>', 'modo_abertura')
# modo_abertura = {'r', 'w', 'a'}

f = open('dados.txt', 'w')
f.write('Texto para introduzir numa linha do
ficheiro')
f.close()
```

Ficheiros: formato CSV

```
# CSV - Comma separated values
# Lista de valores separados por vírgula, ou
outro separador
# Lido pelo Excel como tabela

10,11,12,13,1321 # uma linha com numeros
10,'texto',1'mais texto',12,123,54 # texto +
numeros
```

Ficheiros: formato JSON

```
# JSON – Javascript object notation
# Formato tipo dicionário de Python
# Fácil de converter para dicionário

{“key”:”texto”, “key”:134, ...}

# texto tem que estar dentro de aspas “”
```

Sistema de ficheiros

```
import os

os.path.dirname(path)
# '/foo/bar/item'  --- '/foo/bar '
os.path.basename(path)
# '/foo/bar/item'  --- 'item'
os.path.exists(path) # verifica se caminho
existe
os.path.isfile(path) # verifica se é um ficheiro
os.path.isdir(path) # verifica se é um
directório
os.path.join(path, *paths) # une strings com
separador de directório
```

Questões?

Projecto

- Construir programa de gestão clínica médica:
  - Listar pacientes
  - Inserir, Eliminar pacientes
  - Consultar pacientes:
    - Diagnosticar obesidade através de IMC
    - Diagnosticar diabetes
    - Analisar consumo calórico semanal
  - Estatísticas: min, max, média
    - IMC
    - Idade
    - Peso
    - Ritmo cardíaco

Referências

- https://docs.python.org/3/reference/index.html
- https://docs.python.org/3/library/index.html
- https://www.anaconda.com/distribution/
- https://jupyter.org
- https://www.spyder-ide.org/
- https://learnpythonthehardway.org/book/