Open Journal of Mathematics and Physics | Volume 2, Article 191, 2020 | ISSN: 2674-5747
https://doi.org/10.31219/osf.io/t6x8p | published: 22 Dec 2020 | https://ojmp.org

HN

[white paper]

Diamond Open Access

[aguardando revisão pelos pares]

Caderno Didático em Python

Colaboração Programação Aberta[1]

9 de Março de 2021

Resumo

Apresentamos alguns comandos básicos em python por meio de um ‘‘jupyter notebook’’.

palavras-chave: programação, python, jupyter notebook

A versão mais atualizada deste artigo está disponível em

https://osf.io/t6x8p/download

Introdução

1. O ***Jupyter notebook*** [1] é um *aplicativo web* que integra importantes funcionalidades, como **editor de texto**, **editor latex**, **python** (*códigos* e *output*) [2] em uma *interface gráfica*.

[1]Todos os autores com suas afiliações aparecem no final deste artigo.

Objetivo

2. Este ***caderno*** tem como principal proposta *mostrar o funcionamento de comandos básicos em python* com o *mínimo de explicações* de forma que sua utilização seja compreensível.

3. Espera-se que o *leitor* tenha certa *familiaridade* prévia com linguagem de *programação* tal que possa *entender* os comandos apresentados, simplesmente lendo o **código**, seus **comentários** e o **output** gerado.

Python

4. A versão em PDF do *notebook* está anexada no apêndice deste artigo (após as referências) e o *download* no formato *jupyter* [1] está disponível em [3, 4].

Considerações Finais

5. O passo inicial para se aprender programação de uma linguagem é conhecer sua sintaxe e comandos.

6. A partir de um caminho mais direto, é possível reduzir esforços e acelerar o aprendizado para poder direcionar o foco para as dificuldades inerentes a um algoritmo.

Ciência Aberta

O **arquivo latex** para este artigo, juntamente com outros *arquivos suplementares*, estão disponíveis em [4].

Consentimento

7. Os autores **concordam** com [5].

Como citar este artigo?

8. [6]

Referências

[1] Jupyter. `https://jupyter.org`

[2] Python. `https://www.python.org`

[3] Caderno Didático em Python (*jupyter notebook*). `https://osf.io/9txr4/`

[4] Lobo, Matheus P. "Open Journal of Mathematics and Physics (OJMP)." *OSF*, 21 Apr. 2020. `https://doi.org/10.17605/osf.io/6hzyp`

[5] Lobo, Matheus P. "Simple Guidelines for Authors: Open Journal of Mathematics and Physics." *OSF Preprints*, 15 Nov. 2019. `https://doi.org/10.31219/osf.io/fk836`

[6] Lobo, Matheus P. "Caderno Didático Em Python." *OSF Preprints*, 22 Dec. 2020. `https://doi.org/10.31219/osf.io/t6x8p`

Colaboração Programação Aberta

Matheus Pereira Lobo (autor principal, mplobo@uft.edu.br)[1,2]
https://orcid.org/0000-0003-4554-1372

Sergio Williams Ferreira de Sousa[1]
(programador, wllyvn@gmail.com)

[1]Universidade Federal do Tocantins (Brasil)
[2]Universidade Aberta (UAb, Portugal)

Caderno Didático em Python

March 9, 2021

1 Caderno Didático em Python

1.1 Comentário

[1]:
```
# comentários
# NÃO repetir (nos comentários) o que está claro no programa
```

Use **comentários** para incluir *informações importantes* e não óbvias no código.

Exemplos: + escreva os significados de variáveis matemáticas, + explique por que algumas variáveis estão sendo usadas, + coloque ideias gerais sobre algumas proposições utilizadas.

1.2 PRINT

[2]:
```
# print

print('Olá')

# print line break, pular linha (\n)
print('Teste\n')

# concatenar
print('Math' + 'eus\n')
```

```
Olá
Teste

Matheus
```

[3]:
```
# print várias linhas (aspas tripla)

print('''1
2
3
''')
```

```
1
2
```

```
3
```

[4]:
```
# print formato usando porcentagem

a = 0.123456
b = -1.9876

print('a = %.2f, b = %.2f' % (a,b))        # duas casas decimais
print('a = %.i, b = %.i' % (a,b),'\n')     # toma a parte inteira
```

```
a = 0.12, b = -1.99
a = 0, b = -1
```

[5]:
```
# print format (f)

nome = 'Matheus'
print(f'Meu nome é {nome}')
```

```
Meu nome é Matheus
```

1.3 Matemática Básica

[6]:
```
# normalmente se coloca espaço entre (=,+,-) e não se coloca entre (*,/,**)
# (para facilitar a leitura)
```

[7]:
```
# expoente
2**3
```

[7]:
```
8
```

[8]:
```
#  ==       # igual
#  !=       # diferente
#  >=       # maior ou igual
#  <=       # menor ou igual
#  >        # maior
#  <        # menor
```

1.4 Bibliotecas

[9]:
```
# importar biblioteca de matemática

import math
math.sqrt(4)   # raiz quadrada de 4
```

[9]:
```
2.0
```

[10]:
```
# importar biblioteca de matemática, todas ou apenas algumas funções

from math import *                      # importa todas as funções do módulo␣
↪'math'
from math import sqrt, exp, log, sin    # importa funções específicas do módulo␣
↪'math'
sqrt(4)                                 # raiz quadrada de 4
```

[10]: 2.0

[11]:
```
# importar biblioteca com novo nome

import math as m            # 'm' é agora o nome do módulo 'math'
c = m.cos(m.pi)             # 'c' recebe o cosseno de pi
print('cos(pi) =',c,'\n')
```

```
cos(pi) = -1.0
```

[12]:
```
# importar uma função de uma biblioteca com novo nome

from math import log as ln
l = ln(5)                           # 'l' recebe logaritmo de 5
print(f'ln(5) ={l:.2f}','\n')
```

```
ln(5) =1.61
```

[13]:
```
# importar várias funções de uma biblioteca com novos nomes

from math import sin as s, cos as c, log as ln
v = s(1)*c(2) + ln(3)               # 'v' recebe sen(1)*cos(2) + log(3)
v
```

[13]: 0.7484368002940951

1.5 Miscelânea

[14]:
```
# 'type' mostra o tipo do objeto
```

[15]:
```
type(2.1)
```

[15]: float

[16]:
```
type('abc')
```

[16]: str

[17]:
```
# operações no próprio objeto

soma = 1
print('soma =',soma)

soma += 2                   # equivalente a fazer soma = soma + 2
print('soma =',soma)

subtr = 6
print('\nsubtr =',subtr)

subtr -= 1                  # equivalente a fazer subtr = subtr - 1
print('subtr =',subtr,'\n')

# de forma análoga, utiliza-se o comando '*='
```

```
soma = 1
soma = 3

subtr = 6
subtr = 5
```

[18]:
```
# transforma a string N em número com 'eval'

N = '2'
print(type(N))

N = eval(N)
print(type(N),'\n')
```

```
<class 'str'>
<class 'int'>
```

[19]:
```
# 'eval' é uma função que executa um código python

a = eval('1+2')
b = type(eval('2'))

print(a,'\n',b)
```

```
3
 <class 'int'>
```

1.6 TUPLE

[20]:
```
# tuple

t = (1,2)
print(t)
type(eval('t'))
```

```
(1, 2)
```

[20]:
```
tuple
```

1.7 Conjunto (set)

[21]:
```
# conjunto

C = {1,2,2,3,3,3}
print(C)
type(C)
```

```
{1, 2, 3}
```

[21]:
```
set
```

1.8 STRING (str)

[22]:
```
# string

s = '123'
ss = 'abc'

print(type(s),'\n',type(ss),'\n')
```

```
<class 'str'>
 <class 'str'>
```

[23]:
```
# conta quantas vezes um caracter aparece em uma 'string'

m = 'aaabbc'
m.count(m[0])            # conta os caracteres da posição 0 de 'm'
```

[23]:
```
3
```

[24]:
```
# concatenar caracteres na 'string', pela direita e pela esquerda

b = 'b'
ab = 'a' + b
bc = b + 'c'
```

```
print(ab)
print(bc)
```

```
ab
bc
```

1.9 IF

[25]:
```
# if, elif, else

a = 2
if a == 1:
    print('a = 1')
elif a == 2:
    print('a = 2')
else:
    print('nem 1, nem 2')
```

```
a = 2
```

1.10 FOR

[26]:
```
# 'for' (loop)

val = [5*i for i in range(6)]       # 'for' dentro de uma lista
print(val,'\n')

for v in val:
    print(v)
```

```
[0, 5, 10, 15, 20, 25]

0
5
10
15
20
25
```

1.11 WHILE

[27]:
```
# while

a = 0
while a < 7:
    a += 1
    print('a =',a)
print('')                  # pula uma linha
```

```
a = 1
a = 2
a = 3
a = 4
a = 5
a = 6
a = 7
```

[28]:
```
# 'while' com 'break'

n = 5
while n > 0:
    n -= 1              # equivalente a fazer n = n - 1
    if n == 2:
        break           # interrompe o 'while'
    print(n)
print('Fim do loop','\n')
```

```
4
3
Fim do loop
```

[29]:
```
# 'while' infinito com 'True'

n = 2
while True:   # ou 'while 1:'
    n *= n
    if n > 500: break
    print(n)
print('Fim do loop')
```

```
4
16
256
Fim do loop
```

1.12 INPUT

[30]:
```
# digitar input

a = input('Digite o valor de "a":')
print('a =',a,'\n')
```

```
Digite o valor de "a": 12

a = 12
```

[31]:
```
# digite '1' para continuar ou outra tecla para finalizar

cont = input('Digite 1 para continuar ou outra tecla para finalizar.\n')
if cont == '1':
    print('ok','\n')
else: print('fim','\n')
```

```
Digite 1 para continuar ou outra tecla para finalizar.
 1

ok
```

[32]:
```
# para validar se foi digitado um número natural maior que 0

while True:
    N1 = input('Digite um número natural maior que 0: ')
    print('Validando a entrada...')
    if N1.isdigit():
        N1 = eval(N1)
        if N1 <= 0:
            print('Opção inválida.','\n')
        else: print('ok'); break
    else: print('Opção inválida.\n')
```

```
Digite um número natural maior que 0:  -2

Validando a entrada…
Opção inválida.


Digite um número natural maior que 0:  a

Validando a entrada…
Opção inválida.


Digite um número natural maior que 0:  3

Validando a entrada…
ok
```

1.13 Função

[33]:
```
# função (definição)

def f(x,y):
    return x**2 + y - 1

print('f(2,3) =',f(2,3))
```

```
print('f(2.34,3) =',float(f'{f(2.34,3):.2f}'),'\n')   # arredondamento com 2 casas decimais
```

```
f(2,3) = 6
f(2.34,3) = 7.48
```

[34]:
```
# função lambda (anônima)

# função = lambda var1,var2: return
f1 = lambda x,y: x**2 + y

f1(3,1)
```

[34]:
```
10
```

1.14 Lista

[35]:
```
# definição de uma lista vazia

L = []
print('L =',L,'\n')
```

```
L = []
```

[36]:
```
# lista e sublista

L = [1,2,3,[4,5]]
print('L =',L)
type(L)
```

```
L = [1, 2, 3, [4, 5]]
```

[36]:
```
list
```

[37]:
```
# comprimento da lista

L = [1,2,3]
len(L)
```

[37]:
```
3
```

[38]:
```
# lista (índice)

L = [2,3,5,7]            # L[0] = 2, L[1] = 3, L[2] = 5, ...
L.index(5)               # mostra o índice do elemento '5' da lista
```

[38]:
```
2
```

[39]:
```
# lista (adicionar/remover elementos)

L = [3,2,7]
print('L = ',L)

L = L + [5]              # adicionar elemento no final da lista L
print('L = ',L)

L = [0] + L              # adicionar elemento no início da lista L
print('L = ',L)

L.pop(2)                 # remove o terceiro elemento da lista L
print('L = ',L)

L = ['a'] + L + ['z']    # adiciona um elemento antes e um depois da lista L
print('L = ',L,'\n')
```

```
L =  [3, 2, 7]
L =  [3, 2, 7, 5]
L =  [0, 3, 2, 7, 5]
L =  [0, 3, 7, 5]
L =  ['a', 0, 3, 7, 5, 'z']
```

[40]:
```
# lista (adicionar/remover elementos) usando insert/remove

L = [2,10,12]
L.insert(1,3)      # adiciona na posição 1 o elemento 3
print(L)
L.remove(10)       # remove o elemento 10
L
```

```
[2, 3, 10, 12]
```

[40]:
```
[2, 3, 12]
```

[41]:
```
# manipulando uma lista (adicionar/remover elementos) usando'append'

L = ['a','b']
L.append('c')       # adiciona 'c' no final da lista L
L
```

[41]:
```
['a', 'b', 'c']
```

[42]:
```
# lista (ordem crescente/decrescente) com o 'sort'

L = [3,2,7]
print('L = ',L)
```

```
L.sort()                # coloca a lista ordem crescente
print('L = ',L)

L.sort(reverse = True)  # coloca a lista ordem decrescente
print('L = ',L,'\n')
```

```
L =  [3, 2, 7]
L =  [2, 3, 7]
L =  [7, 3, 2]
```

[43]:
```
# lista (ordem crescente/decrescente) com o 'sorted'

L = [3,2,7]
L = sorted(L)
Lrev = sorted(L,reverse = True)

print(L)
Lrev
```

```
[2, 3, 7]
```

[43]:
```
[7, 3, 2]
```

O 'sort' é mais apropriado para manipular (organizar) o próprio objeto sem retorno. O 'sorted', por outro lado, retorna a lista organizada em ordem crescente/decrescente, não alterando a lista original.

[44]:
```
# lista, substituição

a = [1,2,1,1,3,1]

[0 if x == 1 else x for x in a]    # substitui '1' por '0' na lista 'a'
```

[44]:
```
[0, 2, 0, 0, 3, 0]
```

[45]:
```
# lista filtrada

a = [1,2,1,1,3,1]
b = list(filter(lambda i: i == 1, a))   # o comando 'list' converte seu␣
 ↪argumento em lista

print(a)
b
```

```
[1, 2, 1, 1, 3, 1]
```

[45]:
```
[1, 1, 1, 1]
```

[46]:
```
# string é um objeto iterável

m = 'matheus'
print(m)
print(m[2])

m = m.replace(m[2],'h')     # substitui o 't' por 'h'
print(m)

m = m.replace(m[2],'',2)    # substitui os dois 'h' por nada
print(m,'\n')
```

```
matheus
t
mahheus
maeus
```

[47]:
```
# lista em ordem alfabética

l = ['c','b','a']
print('l =',l)
l.sort()                    # coloca a lista em ordem alfabética
print('l =',l)
l.sort(reverse = True)
print('l =',l,'\n')         # coloca a lista em ordem alfabética reversa
```

```
l = ['c', 'b', 'a']
l = ['a', 'b', 'c']
l = ['c', 'b', 'a']
```

[48]:
```
# 'for' dentro de uma lista

N = [i+1 for i in range(10)]
print('N =',N)

M = [5*i for i in range(10)]
print('M =',M,'\n')
```

```
N = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
M = [0, 5, 10, 15, 20, 25, 30, 35, 40, 45]
```

[49]:
```
# elementos parciais da lista

L = [0,1,2,3,4,5,6,7,8,9]

print(L)        # imprime a lista L
```

```
print(L[:])     # imprime uma cópia da lista L
print(L[:2])    # imprime os dois primeiros elementos da lista L
print(L[2:])    # imprime depois dos dois primeiros elementos da lista L
L
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1]
[2, 3, 4, 5, 6, 7, 8, 9]
```

[49]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

1.15 Dicionário

[50]:
```
# dicionário
# a chave de um dicionário aceita 'string' e 'número'

m = {'a':0,'b':'ok',3:[1,2,3]}

print(m['a'])
print(m['b'])
m
```

```
0
ok
```

[50]:
```
{'a': 0, 'b': 'ok', 3: [1, 2, 3]}
```

[51]:
```
# update
n = {'c':3}

m.update(n)      # atualiza o dicionário 'm', incluindo 'n'
m
```

[51]:
```
{'a': 0, 'b': 'ok', 3: [1, 2, 3], 'c': 3}
```

[52]:
```
# 'get' pode ser usado com um ou dois argumentos; a chave é colocada no␣
 ↪primeiro;
# se não existir a chave, então o segundo argumento é mostrado

print(m.get('b'))
print(m.get('c'))
m.get('d','não tem d')
```

```
ok
3
```

[52]:
```
'não tem d'
```

[53]:
```
# setdefault (é uma combinação do get com o update)

print(m.setdefault('c',4))      # como a chave 'c' existe, ele retorna 3
print(m.setdefault('d',5))      # como a chave 'd' não existe, ele inclui a␣
 ↪chave e o valor no dicionário e retorna o valor
m
```

```
3
5
```

[53]:
```
{'a': 0, 'b': 'ok', 3: [1, 2, 3], 'c': 3, 'd': 5}
```

1.16 WEB

[54]:
```
# importa a biblioteca 'webbrowser'
import webbrowser

# abre o site 'ojmp.org'
webbrowser.open('https://ojmp.org')
```

[54]:
```
True
```