



CAPÍTULO

PYTHON

JPSantos
2021

11. PYTHON

Python é uma linguagem de programação.

<https://www.w3schools.com/python/>

APPRENDRE LE PYTHON #1 ? LES BASES & PREREQUIS

<https://www.youtube.com/watch?v=psaDHhZ0cPs>

11.1. Python instalação

Descarregar o python do site :

<https://www.python.org/downloads/windows/>

Selecionar o instalador e seleccionar a opção “Add Python to PATH”



O programa de instalação:

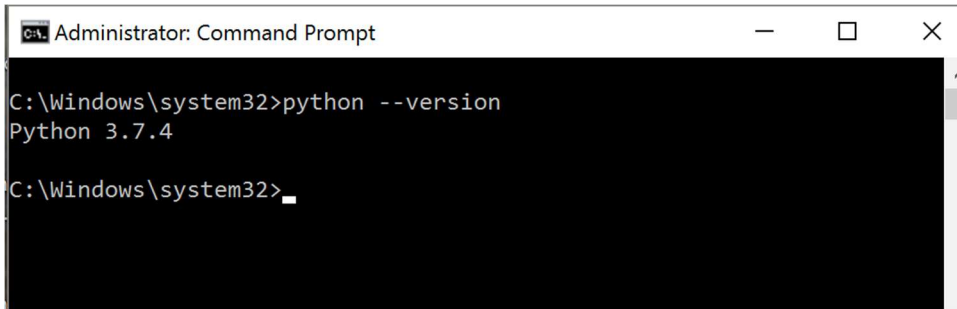
1- instalar no disco duro do computador o ficheiro executável python.exe

c:\users\jpsantos\AppData\Local\Programs\python\...\python.exe

2- atualizar a variável PATH com a localização do programa executável python.exe

3- e instalar um programa chamado “pip” que permite entre outras coisas actualizar/descarregar novos módulos/bibliotecas em Python.

Para testar se o python ficou correctamente instalado, abra uma janela de comandos de linha em modo administrador, escreva *python --version*



```
Administrator: Command Prompt
C:\Windows\system32>python --version
Python 3.7.4
C:\Windows\system32>_
```

Depois edite um ficheiro de texto com a extensão .py, com uma linha apenas:

print('Hello')

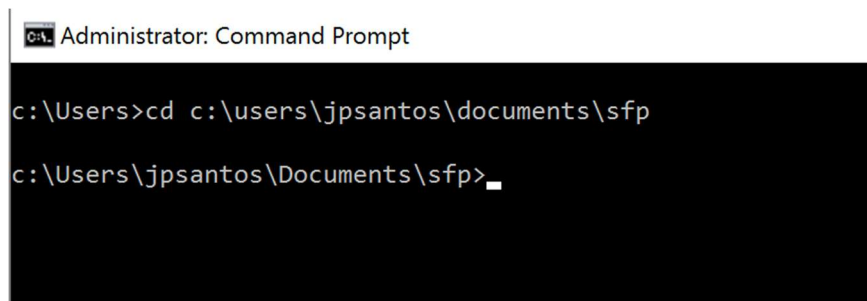
Guarde esse ficheiro com a extensão .py, por exemplo “teste.py” no directório:

c:\users\jpsantos\documents\SFP\teste.py

Execute esse ficheiro/programa em python, com uma só linha, numa janela de comandos de linha:

primeiro deve mudar o directório de trabalho para o directório onde gravou o ficheiro .py fazendo cd - change directory:

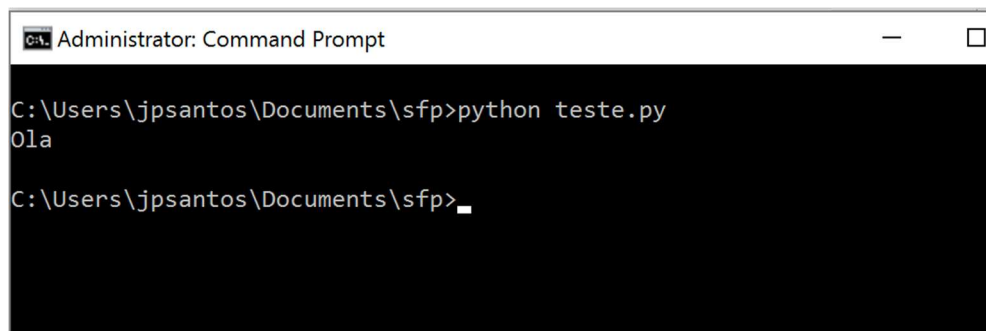
> cd *c:\users\jpsantos\documents\SFP*



```
Administrator: Command Prompt
c:\Users>cd c:\users\jpsantos\documents\sfp
c:\Users\jpsantos\Documents\sfp>_
```

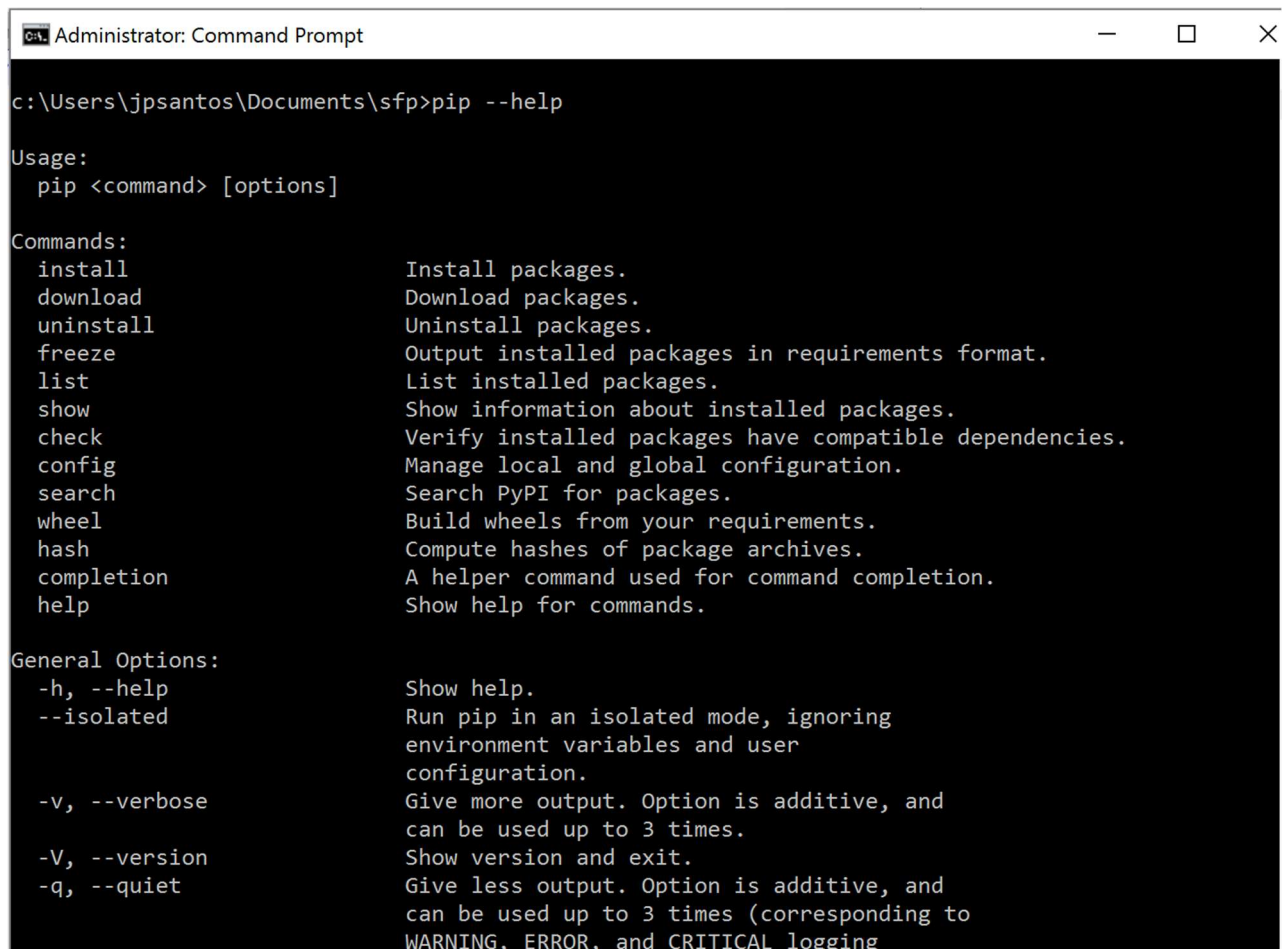
O programa executável “python.exe” irá abrir o ficheiro .py e executar/interpretar a(s) instruções que contém.

> python teste.py



```
C:\Users\jpsantos\Documents\sfp>python teste.py
Ola
C:\Users\jpsantos\Documents\sfp>
```

Existem bibliotecas já desenvolvidas, com código que pode incluir e reutilizar nos seus programas. Pode usar o programa “pip” para descarregar da internet essas bibliotecas, módulos de python. A figura seguinte apresenta os comandos que o programa “pip” reconhece:



```
c:\Users\jpsantos\Documents\sfp>pip --help

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze            Output installed packages in requirements format.
  list              List installed packages.
  show              Show information about installed packages.
  check             Verify installed packages have compatible dependencies.
  config            Manage local and global configuration.
  search            Search PyPI for packages.
  wheel             Build wheels from your requirements.
  hash              Compute hashes of package archives.
  completion        A helper command used for command completion.
  help              Show help for commands.

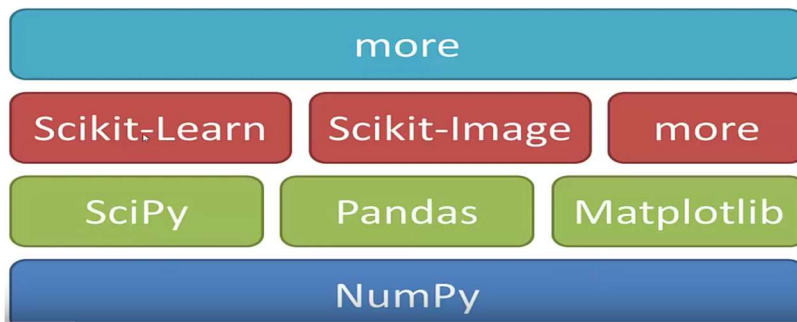
General Options:
  -h, --help        Show help.
  --isolated         Run pip in an isolated mode, ignoring
                    environment variables and user
                    configuration.
  -v, --verbose     Give more output. Option is additive, and
                    can be used up to 3 times.
  -V, --version     Show version and exit.
  -q, --quiet       Give less output. Option is additive, and
                    can be used up to 3 times (corresponding to
                    WARNING, ERROR, and CRITICAL logging
```

Na janela de comandos de linha (em modo administrador), “cmd”, pode usar o programa PIP para instalar os vários módulos do Python:

```
> pip install numpy
> pip install pandas
```

Na secção 12.2 será apresentada a linguagem Python, Built-in modules
<https://docs.python.org/3/py-modindex.html>

Nas secções 12.3 a 12.9 serão apresentados os módulos Python adicionais:



pesquisar em pypi.org módulos desenvolvidos por outros programadores.

11.2. Python Linguagem

Os ficheiros de texto com instruções em Python são executados/interpretados por um programa executável previamente instalado no computador chamado Python.exe

11.2.1. Variáveis

As variáveis em Python, como noutras linguagens, podem conter dados e têm uma existência real, mas efémera, em zonas de memória RAM do computador.

O nome das variáveis de Python tem de começar por uma letra ou pelo carácter “_”. Letras maiúsculas ou minúsculas são diferentes para o Python.

Tipos de variáveis

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

Conversores

```
x = str("Hello World")
x = int(20)
x = float(20.5)
x = complex(1j)
x = list(("apple", "banana", "cherry"))
x = tuple(("apple", "banana", "cherry"))
x = range(6)
x = dict(name="John", age=36)
x = set(("apple", "banana", "cherry"))
x = frozenset(("apple", "banana", "cherry"))
x = bool(5)
x = bytes(5)
x = bytearray(5)
x = memoryview(bytes(5))
```

exemplo:

```
AnoNasc = "1999"           # variável do tipo String
AnoAtual = 2019            # variável do tipo Inteira
AnosDecorridos = AnoAtual - AnoNasc    # esta operação não pode ser realizada
AnosDecorridos = AnoAtual - int(AnoNasc)  # é necessário converter a string num número
```

exemplo:

Edite um ficheiro de texto com o nome “teste.py”, com o seguinte conteúdo:

Em python os comentários começam por cardinal

```
a="Ola"           # Cria a variável "a" e atribui-lhe a string 'Ola'
b='123'          # Cria a variável "b" e atribui-lhe a string '123'
c=3.25           # Cria a variável "c" e atribui-lhe o valor float 3,25
d= 3             # Cria a variável "d" e atribui-lhe o valor inteiro 3
e= 2E-3          # Cria a variável "e" e atribui-lhe 2x10 -3
f= 2j            # Cria variável "f" e atribui-lhe o número complexo 2j
print(a)
print(b)
print(c)
print(d)
print(e)

print("A"*10)     # imprime 10 'A' seguidos      AAAAAAAAAA
s= "Joao"
print("Nome:" + s) # imprime      Nome:Joao

first='Jose'
last='Santos'
msg=f"{first} {last} é professor "
print(msg)        # imprime      Jose Santos é professor
```

O exemplo seguinte, declara três variáveis e atribui-lhes o valor 0

```
x=y=z=0
print(x)      # imprime "0"
print(y)      # imprime "0"
print(z)      # imprime "0"
```

O exemplo seguinte, declara três variáveis e atribui-lhes os valores “um”, “dois” e “tres”

```
s1,s2,s3="um","dois","tres"
print(s1)      # imprime "um"
print(s2)      # imprime "dois"
print(s3)      # imprime "tres"
```

O exemplo seguinte :

```
L=['a','b','c','d','e'] # é igual declarar e inicializar a variável destas três formas
L='abcde'               #  “
L="abcde"               #  “
print(L[0])             # visualiza o conteúdo do primeiro elemento, neste caso 'a'
print(L[4])             # visualiza o conteúdo do quinto elemento, neste caso 'e'
print(L[-1])            # visualiza o conteúdo do último elemento, neste caso 'e'
print(L[-2])            # visualiza o conteúdo do penúltimo elemento, neste caso 'd'
print(L[0:2])           # visualiza o conteúdo, neste caso 'ab'
print(L[0:-2])          # visualiza o conteúdo, neste caso 'abc'
```

Uma variável pode ser declarada dentro de uma função, e nesse caso é uma **variável local**, só pode usada dentro dessa função, a zona da RAM do computador correspondente só está reservada para ela enquanto o código dessa função estiver a ser executado.

Exemplo: https://www.w3schools.com/python/python_variables.asp

```
x = "awesome"                # variável Global

def myfunc():
    x = "fantastic"          # variável Local
    print("Python is " + x)   # imprime a variável Local "Python is fantastic"

myfunc()
print("Python is " + x)       # imprime a variável Global "Python is awesome"
```

Strings - funções

```
Nome='josE Paulo santos'
print(Nome.capitalize())    # imprime Jose paulo santos
print(Nome.find('o'))        # imprime 1
print(Nome.replace('j','p')) # imprime pose Paulo santos
print(Nome.count('s'))       # imprime 3
print(Nome.islower())        # imprime False
print(Nome.isnumeric())      # imprime False
print(Nome.lower())          # imprime jose paulo santos
print(Nome.upper())          # imprime JOSE PAULO SANTOS
r=Nome.split(' ')            # divide a string em 3 substrings, o separador é o espaço
print( r[0] )                # imprime josE
print( r[1] )                # imprime Paulo
print( r[2] )                # imprime santos

print( len(Nome) )           # imprime 17
```

Maths - funções

<https://docs.python.org/3.0/library/math.html>

```
x= -3.14
print( round(x) )
print( abs(x) )

import math

print( math.acos(1) )        # 0.0
print( math.acosh(1) )       # 0.0
print( math.cos(3.14) )      # -1
print( math.cosh(1) )        # 1.54
print( math.tan(3.14/4) )    # 1
print( math.tanh(3.14/4) )   # 0.65
print( math.sqrt(2) )        # 1.41
print( math.degrees(3.14) )  # 180
```

Entrada de dados a partir do teclado - input

```
nome= input("User: ")          # a função input permite ao utilizador introduzir uma string a partir do teclado
chave= input("Password: ")
print( "O nome é " + nome + " e a password é " + chave)
```

11.2.2. Listas e Dicionários

O python utiliza essencialmente dois tipos de estruturas, as listas e os dicionários. Uma lista é composta por itens organizados de forma linear, sendo que cada item pode ser acedido através de um índice que representa a posição desse item na lista. De notar que os índices começam em “zero”. Listas são estruturas de dados bastante flexíveis, pelo que podem conter qualquer tipo de elemento. Por exemplo, a mesma lista pode conter variáveis numéricas, booleanas ou strings. Existe também a possibilidade de uma de um dos elementos da lista ser uma lista.

Para a manipulação destas estruturas, existem diversos operadores que podem ser utilizados, como por exemplo:

```
List=['0']
print(List)

List.append('1')
List.append('2')
List.append('3')          # adiciona um novo elemento no final da lista.
print(List)

List.pop(1)               # remove o elemento na posição especificada. Se nada for especificado,
print(List)               # remove o elemento no final da lista.
List.insert(1, '1')       # insere novo valor na posição especificada
print(List)

print(List.count('1'))    # retorna o número de vezes que um operador aparece na lista.
print(len(List))         # retorna o número de elementos na lista.
```

Outros operadores em: https://www.3schools.com/python/python_lists.asp

Os dicionários, são estruturas mais complexas. Este contém chaves e valores para as mesmas. Por exemplo, podemos ter um dicionário com as chaves “frutas” e “árvores”. Cada uma destas chaves pode ter uma lista de valores. O código para a criação deste dicionário seria algo do género:

```
Dictionary={}              # inicialização do dicionário
Dictionary['arvores'] = ['macieira','pereira']    # criação do elemento 'arvores'
Dictionary['frutas'] = ['pera','ananas']          # criação do elemento 'ananas'
```

O objeto criado seria do tipo Dictionary:{ 'arvores' : ['macieira','pereira'] , 'frutas' : ['pera','ananas'] }


```

a = [] # initializes list
b = {} # initializes dict
b['frutas'] = ['laranja', 'morango', 'ananás']
b['arvores'] = ['macieira', 'pereira']
a.append(1) # add new value in last position of list
a.append(2)
a.insert(1, 'new value') # insert new value in specified position
print(a)
a.pop(1) # removes the value in specified position or "del a[1]" also works, clear empties the list
print(a)
print(b)
print(b['frutas'][0])
b['frutas'][0]='limão'
print(b)
b['frutas'].append('melão')
print(b)
c = [1, 2, 3]
print(a)
print(a.count(1)) # counts how many times the element appears in the list
print(a+c) # concatenate lists

```

11.2.3. Instruções de controlo

A “linguagem python” é bastante simples e elegante, quando comparada com outras linguagens como por exemplo C# ou C++. Não são necessários “;” no final de cada linha de código, nem chavetas “{ }” a delimitar ciclos “for”, “while” ou comparações “if”. O código apenas tem de estar indentado corretamente, para ser interpretado pelo python.

FOR

Na figura abaixo mostra-se um exemplo de um ciclo “for” em python. Ao contrário de outras linguagens, é possível iterar todos os elementos de uma estrutura (uma lista, neste caso), sem recorrer a indexação. Contudo é possível fazer o mesmo ciclo iterando os índices do elemento,

```

A = ['a', 'b', 'c', 'd', 'e']

for elemento in A:
    print(elemento)
print("_____")
for i in range(len(A)):
    print(A[i])

```

```

L=['a','b','c','d','e']
print(L[0])           # visualiza o conteúdo do primeiro elemento, neste caso 'a'
print(L[4])           # visualiza o conteúdo do quinto elemento, neste caso 'e'

```

```

for i in range(len(L)):
    print( L[i])       # visualiza todos os elementos de L: a,b,c,d,e

```

```

for elemento in L:
    print( elemento)  # visualiza todos os elementos de L: a,b,c,d,e

```

```

for elemento in range(5, 10, 2):
    print( elemento)  # visualiza todos os elementos de L: a,b,c,d,e

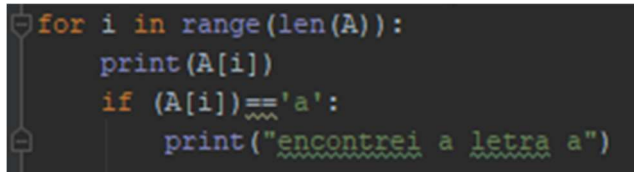
```

IF

A “hierarquia” de código funciona de acordo com a indentação do mesmo, como mostra a figura abaixo.

```
i=10
if i > 18:                                # (i > 18) and (i < 66) # or # not # == # != # >=
    print("É maior de idade")
else:
    print("É um puto ")
```

A operação “if” é efetuada para cada iteração do ciclo for.



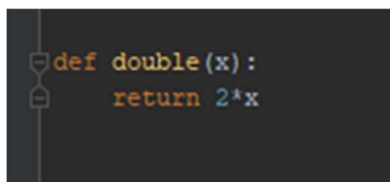
```
for i in range(len(A)):
    print(A[i])
    if (A[i])=='a':
        print("encontrei a letra a")
```

Exemplo “Conversor de unidades de peso”

```
peso = int( input('Introduza o seu peso: '))
unidades= input(' (L)bs ou (K)g: ')
if unidades.upper() == 'L':
    convertido = peso * 0.45
    print(f' O seu peso é {convertido} Kilos')
else:
    convertido = peso / 0.45
    print(f' O seu peso é {convertido} Lbs')
```

11.2.4. Funções

Para criar funções, deve obedecer-se à sintaxe da figura abaixo, onde está exemplificado um método que retorna o dobro do parâmetro passado.



```
def double(x):
    return 2*x
```

É possível criar uma função de duas formas. Por exemplo a função que retorna o quadrado de x $f(x) = x^2$

```
def f(x):                                # criação da função f(x)
    return x**2                          # termina a função e retorna o valor x**2
print(f(2))

f=lambda x: x**2                         # outra forma de criar a função f(x)
print(f(2))
```

11.3. NumPy Library

NumPy: As described in <https://docs.scipy.org/doc/numpy/user/whatIsnumpy.html>

Library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

<https://www.youtube.com/watch?v=xECXZ3tyONo>

<http://cs231n.github.io/python-numpy-tutorial/>

<https://docs.scipy.org/doc/numpy/reference/>

- Numpy Array, Multi-dimensional array, Reshape, Slices, One-Hot Encoding, Reduction Operations

Como já vimos, os objetos python podem ser listas ou dicionários, o que difere por exemplo do Matlab em que cada estrutura é composta por matrizes. Contudo as funções de matlab e ter matrizes como estruturas de dados, pode ser útil em algumas circunstâncias.

Por essa razão, o python tem a biblioteca Numpy, que permite ter grande parte das funcionalidades do Matlab.

Esta biblioteca pode ser instalada na linha de comandos na janela “cmd” ou na janela de comando do pycharm,

```
>pip install numpy
```

Abaixo encontram-se algumas funcionalidades típicas do matlab que podem ser utilizadas no python com esta biblioteca:

código exemplo para **criar** matrizes :

<http://cs231n.github.io/python-numpy-tutorial/#python-basic>

```
import numpy as np

v = np.array([1,2,3])          # Cria um array/vector com três elementos
print(v)                      # imprime o conteúdo do array "[1 2 3]"

x = np.array([[1,2], [3,4]])   # Cria um array de duas dimensões 4 elementos
print(x)                      # imprime o conteúdo da matriz
                                # "[[1 2]
                                #      [3 4]]"
print(x.T)                    # imprime a transposta da matriz
                                # "[[1 3]
                                #      [2 4]]"

x= np.arange(-1,11)           # Cria um array [-1,0,1,2,3,4,5,6,7,8,9,10]
print(x)

y= np.exp(-x/3)               # Cria o array y = e(-x/3) , para todos os elementos de x
print(y)                      # [1.39, 1., 0.71, ..., 0.035]
```

código exemplo para **somar** matrizes :

```
import numpy as np

x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = x + v
print(y)                                     # Somou o vector V a cada linha da matriz X
                                           # Imprimiu
                                           #      "[[ 2  2  4]"
                                           #      [ 5  5  7]"
                                           #      [ 8  8 10]"
                                           #      [11 11 13]]"
```

O módulo numPy tem imensas funcionalidades

<https://docs.scipy.org/doc/numpy/reference/>

Algumas das funções mais interessantes para algoritmia são funções para a geração de números aleatórios, como por exemplo:

`numpy.random.uniform(low,high,size)`: cria um vetor aleatorio de tamanho “size”, de acordo com a distribuição normal entre os valores “low” e “high”.

<https://numpy.org/doc/1.17/user/basics.types.html>

<https://www.youtube.com/watch?v=xECXZ3tyONo>

<http://cs231n.github.io/python-numpy-tutorial/>

11.4. Pandas Library

<https://www.youtube.com/watch?v=e60ltwlZTKM>

<https://pandas.pydata.org>

https://machinelearningmastery.com/scientific-functions-in-numpy-and-scipy/?utm_source=drip&utm_medium=email&utm_campaign=Massaging+data+using+pandas&utm_content=Massaging+data+using+pandas

Pandas (data analysis, data frame de duas dimensões, linhas e colunas)

Na janela de comandos de linha “cmd” instalar o módulo “Panda”

```
> pip install pandas
```

11.4.1. Exemplo pandas I - DataFrame()

Código “pandas_weather.py”:

```
import pandas as pd

df = pd.DataFrame(                # Cria um objecto do tipo DataFrame
[
    ['Jan',58,42],                # Cria uma linha da DataFrame
    ['Fev',68,42],                # outra linha
    ['Dec',48,42],                # outra linha
],
index = [0,1,2],                 # numera as linhas da DataFrame
columns = ['month','avg_high','avg_low'] # Cria o cabeçalho da DataFrame
)

print(df)                        # imprime a tabela criada, com cabeçalho e número da linha

# month      avg_high  avg_low
# 0 Jan        58        42
# 1 Feb        68        42
# 2 Dec        48        42
```

Na janela de comandos de linha “cmd”, correr o programa em python

```
> python pandas_weather.py
```

11.4.2. Exemplo panda II - Gravar e ler ficheiro: `to_csv()`, `to_excel()`, `read_csv()`, `read_excel()`

Código “pandas_weather.py”:

```
> pip install pandas
> pip install openpyxl
> pip install xlrd
```

```
import pandas as pd
df = pd.DataFrame(           # Cria um objecto do tipo DataFrame
[
    ['Jan',58,42],           # Cria uma linha da DataFrame
    ['Fev',68,42],
    ['Dec',48,42],
],
index = [0,1,2],
columns = ['month','avg_high','avg_low']      # Cria o cabeçalho da DataFrame
)
print(df)      # imprime a tabela criada, com cabeçalho e número da linha (index)
```

#	month	avg_high	avg_low
# 0	Jan	58	42
# 1	Fev	68	42
# 2	Dec	48	42

```
df.to_csv('modified.csv',index=False)  # guarda os dados do objecto panda “df”
                                         # no ficheiro de texto ‘modified.csv’
```

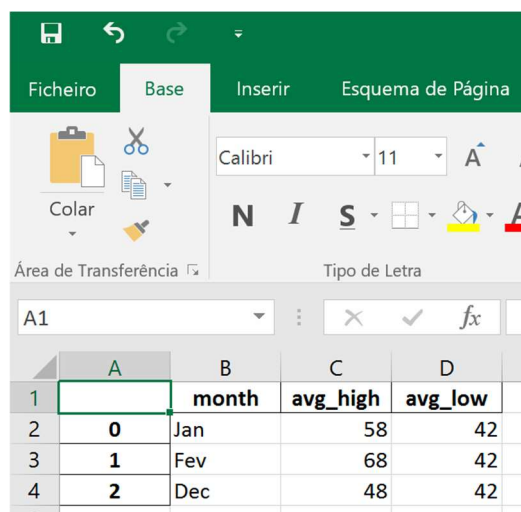
```
dd=pd.read_csv('modified.csv')          # lê o ficheiro de texto ‘modified.csv’,
                                         # cria e inicializa o objecto do tipo panda ,“dd”
print(dd)
```

```
df.to_excel('modified.xlsx')            # guarda o dados no ficheiro ‘modified.xlsx’
```

```
dg=pd.read_excel('modified.xlsx',index_col=0) # Lê o ficheiro , cria o objecto “dg” e inicializa-o
print(dg)
```

Na janela de comandos de linha “cmd” correr o programa em python

```
> python pandas_weather.py
```



	A	B	C	D
1		month	avg_high	avg_low
2	0	Jan	58	42
3	1	Fev	68	42
4	2	Dec	48	42

https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html

11.4.3. Exemplo panda III - df.head() df.tail(3), dtypes, columns, index, values

Código “pandas_weather.py”:

```
import numpy as np
import pandas as pd

filename="fremont_weather.txt"
df=pd.read_csv(filename)
print( df)                # imprime toda a tabela
print( df.head() )        # imprime as 5 primeiras linhas
print( df.tail(3) )       # imprime as últimas 3 linhas da tabela
print( df.dtypes )        # imprime o tipo de dados de cada coluna da tabela: month object avg_high int64
print( df.index )         # imprime os índices: start=0, stop=12 , step=1
print( df.columns )       # imprime o nome das colunas: 'month' 'avg_high' ...
print( df.values )        # imprime os valores da tabela, excepto o nome das colunas
print( df.sort_values('months', ascending=False)) # imprime os valores da tabela com as linhas ordenadas
print( df.months )        # imprime as várias linhas da tabela apenas para a coluna 'months'
print( df[2:4] )          # imprime as linhas da tabela apenas para as colunas 2 e 3 ( a 4 não aparece)
print( df.iloc[3:5,[0:3]] ) # imprime as linhas 3 e 4 da tabela, apenas para as colunas 0,1 e 2
print( df[df.avg_highs > 58] ) # imprime as linhas com avg_highs > 58
```

Na janela de comandos de linha “cmd” execute o programa em python
> python pandas_weather.py

11.4.4. Exemplo panda IV - Estatística

Código:

```
import statistics as st
```

st.count()	Number of non-null observations
st.sum()	Sum of values
st.mean()	retorna a media dos valores do objecto de estatistica , st
	<i>import statistics as st</i>
	<i>nums=[1,2,3,5,7,9]</i>
	<i>pd.mean(nums)</i> <i># retorna 4.5</i>
st.median()	Median of Values
st.mode()	Mode of values
st.std()	Standard Deviation of the Values

1. For each observation, subtract each observation from the average

2. Square each difference

3. Sum up all the differences

4. Divide sum by the total number of observations minus 1

5. Square root the result

Greek letter sigma: used to represent standard deviation

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

st.min()	Minimum Value
st.max()	Maximum Value
st.abs()	Absolute Value
st.prod()	Product of Values
st.cumsum()	Cumulative Sum
st.cumprod()	Cumulative Product

Existem inúmeras funções do módulo “Panda”

<https://pandas.pydata.org/pandas-docs/stable/>

11.5. Python Matplot

MathPlotLib (para criar gráficos de duas dimensões)

<https://matplotlib.org/examples/>

<https://www.youtube.com/watch?v=a9UrKTVEeZA>

> pip install matplotlib (instalar como administrador)

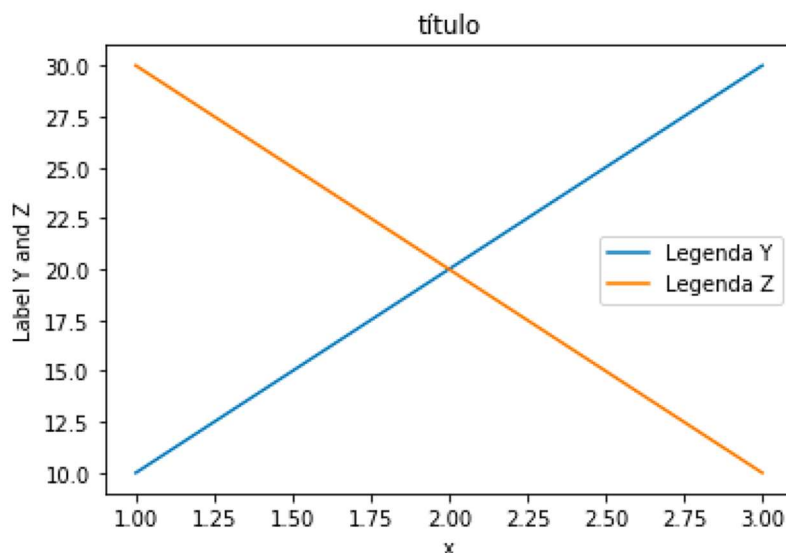
11.5.1. Exemplo MatPlotLib - gráfico com título, dois labels, duas legendas

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

x=[1,2,3]
y=[10,20,30]
z=[30,20,10]

plt.plot(x,y)
plt.plot(x,z)
plt.title('título')
plt.xlabel("x")
plt.ylabel("Label Y and Z")
plt.legend(["Legenda Y", "Legenda Z"])

plt.show()
```



11.5.2. Exemplo Matplotlib - ler os dados num ficheiro e criar o gráfico

Ler os dados num ficheiro e criar o gráfico

```
import numpy as np
import pandas as pd

file='teste.xlsx'
df=pd.read_excel(file)

from matplotlib import pyplot as plt

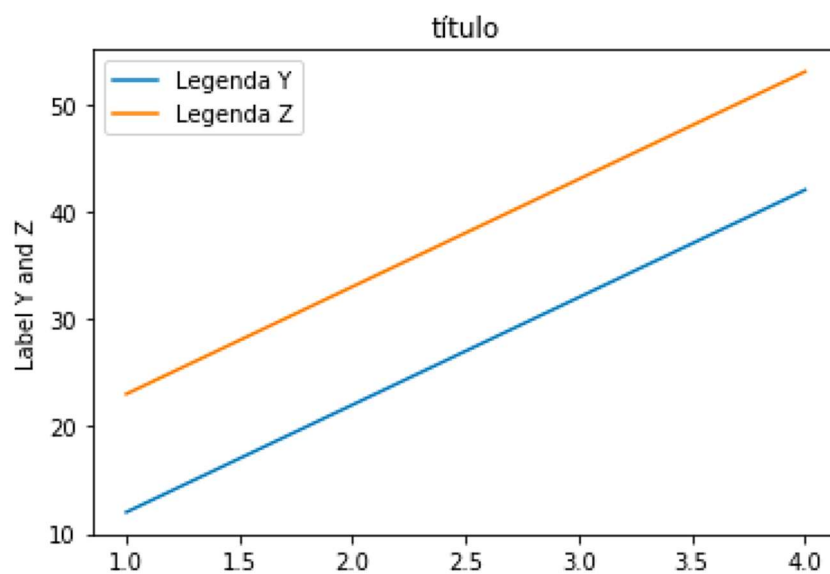
x=df.a
y=df.b
z=df.c

plt.plot(x,y) # também seria possível fazer plt.plot(x,y,'o', x, z, '--') das duas series numa só ...
plt.plot(x,z)
plt.title('título')
plt.xlabel("x")
plt.ylabel("Label Y and Z")
plt.legend(["Legenda Y", "Legenda Z"])

plt.show()
```

Ficheiro excel “teste.xlsx”

a	b	c	d
1	12	23	14
2	22	33	24
3	32	43	34
4	42	53	44



11.6. Python MySQL

https://www.w3schools.com/python/python_mysql_getstarted.asp

> pip install **pymysql**

Considere que tem uma base de dados com nome “reservatorio” e uma tabela “controloreservatorio”, com as colunas e os valores presentes na figura:



	Y0	Y1	Y2
1	0	0	0

Se executar o exemplo:

```
import pymysql

con = pymysql.connect(user='root', password="", host='localhost', database='reservatorio')
cur=con.cursor()
cur.execute("SELECT * from controloreservatorio")
rows=cur.fetchall()
print(rows[0])
cur.close()
con.close()
```

Obtém os valores da primeira linha da tabela
(1, 0, 0, 0)

Outro exemplo, com o módulo **mysql-connector**

<https://dev.mysql.com/downloads/connector/python/>

> pip install mysql-connector

```
import mysql.connector

mydb = mysql.connector.connect(host="localhost", user="root", passwd="", database="reservatorio")
print(mydb)

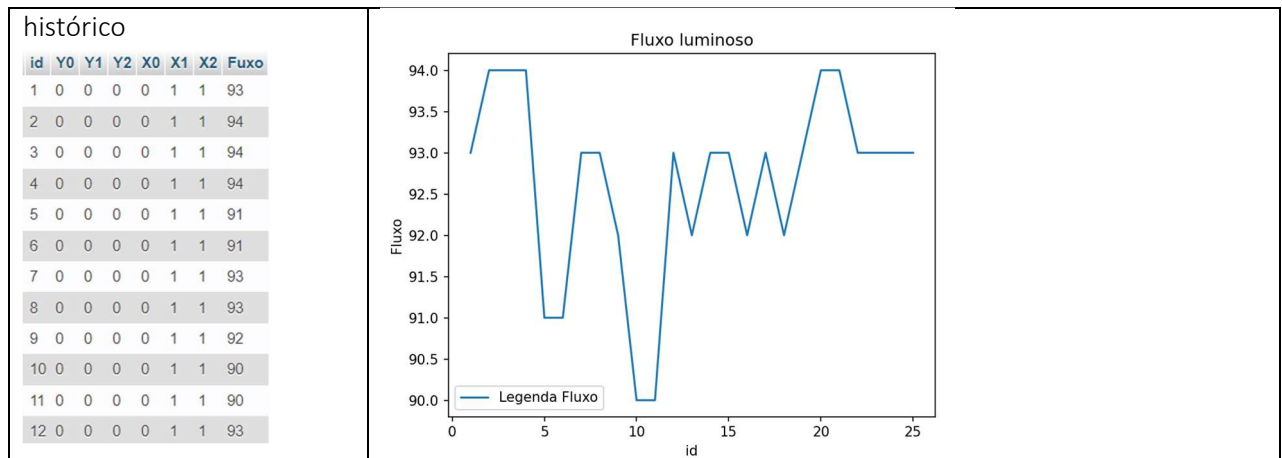
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM controloreservatorio")

myresult = mycursor.fetchone()    # Seleciona uma linha/registo da resposta
print(myresult)

myresult = mycursor.fetchall()    # Seleciona todas as linhas/registos da resposta
for x in myresult:
    print(x)
```

Trabalho

Pretende-se visualizar num gráfico “matplotlib” os valores do fluxo luminoso registados na tabela “historico” da base de dados “reservatório”.



Solução

```
import pymysql
```

```
con = pymysql.connect(user='root', password="", host='localhost', database='reservatorio')
```

```
cur=con.cursor()
```

```
cur.execute('SELECT * from historico')
```

```
rows=cur.fetchall()
```

```
x=[]
```

```
y=[]
```

```
for col in rows:
```

```
    print(col[0])
```

```
    x.append(col[0])
```

```
    y.append( col[7])
```

```
cur.close()
```

```
con.close()
```

```
from matplotlib import pyplot as plt
```

```
plt.plot(x,y)
```

```
plt.title('Fluxo luminoso')
```

```
plt.xlabel('id')
```

```
plt.ylabel('Fluxo')
```

```
plt.legend(["Legenda Fluxo"])
```

```
plt.show()
```

11.7. Python - Request Library

Permite efetuar comunicação com vários servidores e APIs. A partir da URI de uma API, faz o pedido http ao servidor especificado e retorna a resposta do mesmo.

Instalação:

```
> pip install requests
```

Utilização:

```
Import requests
```

https://www.w3schools.com/python/ref_requests_post.asp

https://www.w3schools.com/python/module_requests.asp

https://www.w3schools.com/python/ref_requests_response.asp

Neste exemplo o Apache está ativo e recebe o pedido “request”.

No directório “htdocs” existe um ficheiro “calc.php”

Ficheiro em Python.

<pre>import requests def get(): url = "http://localhost/calc.php?X=10&Y=20" response = requests.get(url) print(response.url) print(response.text) print(response.request) print(response.status_code) return response get()</pre>	<p>Resposta na janela CMD</p> <pre>http://localhost/calc.php?X=10&Y=20 <html> <body> 10+20=30 </body> </html> <PreparedRequest [GET]> 200</pre>
---	--

Calc.php

```
<html>
<body>

<?php
    $Z=$_GET['X']+$_GET['Y'];
    echo $_GET['X'],"+",$_GET['Y'], "=", $Z;
?>

</body>
</html>
```

Outro exemplo, é dada uma função que permite a comunicação com um serviço de meteorologia (<https://openweathermap.org/>). Para conseguir usufruir do serviço é necessário criar uma conta (gratuita) e subscrever um serviço. De seguida, o utilizador recebe um código que é passado na URI aquando do pedido. Este código serve para monitorizar e controlar os pedidos feitos ao servidor. No exemplo abaixo é mostrada uma função que permite obter dados meteorológicos para um determinado ponto (latitude e longitude).

```
def get_weather(lat, lon):
    url = "http://api.openweathermap.org/data/2.5/weather?lat=" + str(lat) + "&lon=" + str(lon)+
        "&APPID=insert_APICredentials"

    response = requests.get(url)
    data = response.json()
    return data
```

11.8. Python SkLearn

Machine learning

> pip install sklearn

- Adquirir os dados.

```
import pandas as pd
music_data = pd.read_csv(musicas1.csv)
```

id, idade, genero (masculino-0/feminino-1), musica (género de música)

idade	género	musica
20	1	Hipop
23	1	Hipop
25	1	Hipop
26	1	Jazz
29	1	Jazz
30	1	Jazz
31	1	Classical
33	1	Classical
37	1	Classical
20	0	Dance
21	0	Dance
25	0	Dance
26	0	Acoustic
27	0	Acoustic
30	0	Acoustic
31	0	Classical
34	0	Classical
35	0	Classical

- Tratar os dados.

remover dados duplicados, nulos, etc.

Dividir os dados (dividir as colunas, input e output).

Uma tabela fica com as colunas id, age, gender, e sem a coluna género musical (genre) - X

Outra tabela fica com as colunas id, e genre (género musical) - Y

A tabela X contém os dados de entrada para o algoritmo, a tabela Y contém os resultados.

```
x = music_data.drop(columns=['musica']) # remove a coluna 'musica', não altera o ficheiro original
# X passa a conter id, idade, genero
y = music_data['musica'] # Y passa a conter a tabela id, musica (género musical)
```

- Treinar o algoritmo

**** Create model**

**** Train model**

**** Make prediction**

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()           # create model
model.fit(x,y)                             # Train model

# pergunta qual música preferida de uma mulher de 21 anos e de um homem de 20 anos
predictions = model.predict([ [21,1], [20,0] ])

# resposta 'HipHop', 'Dance', respectivamente.
predictions
```

Exemplo completo:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
music_data=pd.read_csv('musicas1.csv')
print(music_data)
x= music_data.drop(columns=['musica'])
y= music_data['musica']
model= DecisionTreeClassifier()
model.fit(x,y)
predictions=model.predict([[21,1],[20,0]])
print(predictions)
```


11.9. Python - Pulp

De acordo com o projecto PuLP

PuLP is an LP modeler written in python. PuLP can generate MPS or LP files and call GLPK[1], COIN CLP/CBC[2], CPLEX[3], and GUROBI[4] to solve linear problems.

PuLP: A Linear Programming Toolkit for Python

<https://optimization-online.org/2011/09/3178/>

Para instalar o módulo, a partir do terminal:

```
> pip install PuLP
```

Documentation is found on <https://www.coin-or.org/PuLP/>.

A comprehensive wiki can be found at

<https://www.coin-or.org/PuLP/>

Use `LpVariable()` to create new variables.

To create a variable $0 \leq x \leq 3$ >>> `x = LpVariable("x", 0, 3)`

To create a variable $0 \leq y \leq 1$ >>> `y = LpVariable("y", 0, 1)`

```
import pulp
```

```
prob=pulp.LpProblem("Giapetto",pulp.LpMaximize)
```

```
x1=pulp.LpVariable("x1",lowBound=0)
```

```
x2=pulp.LpVariable("x2",lowBound=0)
```

```
prob+=20*x1+30*x2
```

```
prob+=1*x1+ 2*x2 <= 100
```

```
prob+=2*x2+1*x1 <= 100
```

```
status=prob.solve()
```

```
pulp.LpStatus[status]
```

```
print(status)
```

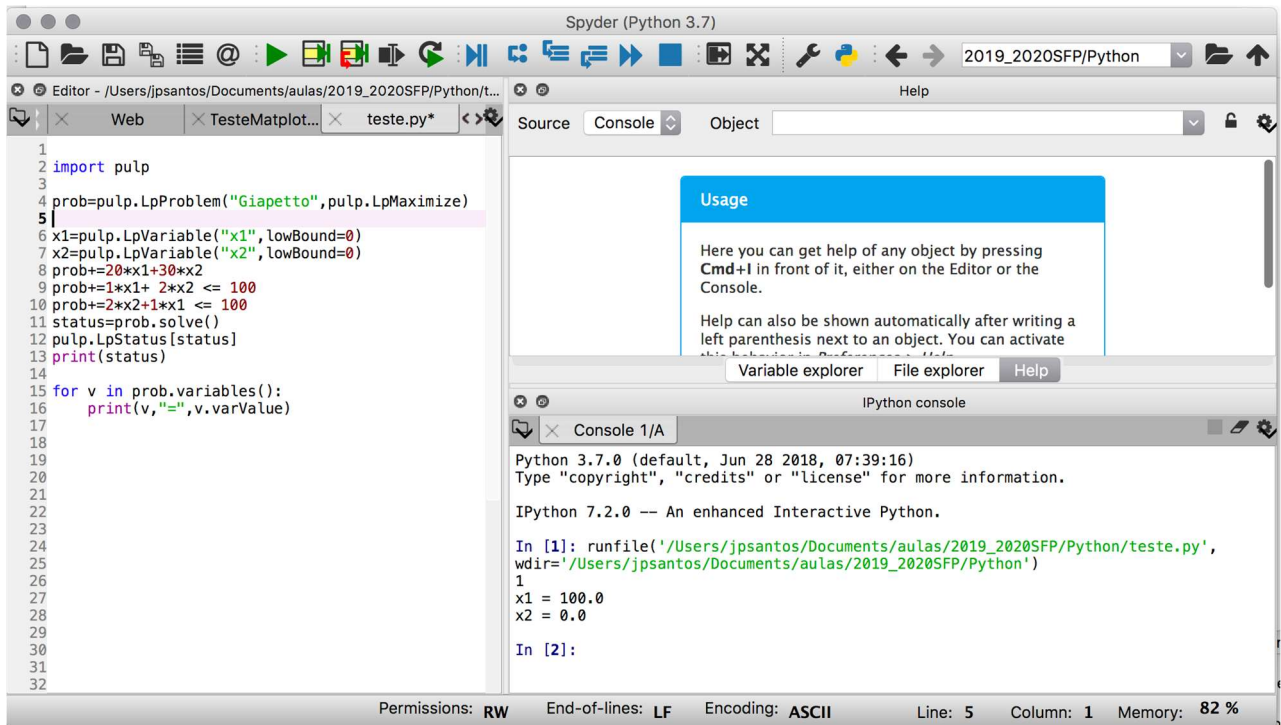
```
for v in prob.variables():
```

```
    print(v,"=",v.varValue)                # imprime x1= 100.0  x2= 0.0
```

O resultado é :

x1 = 100.0

x2 = 0.0



Trabalho de casa

Implementar em Python o caso seguinte:

Um exemplo de Branch & Bound para “job shop”, na tarefas para uma máquina.

Pesquisa de Operações 09D: Problema de Agendamento de Oficina de Trabalho

<https://www.youtube.com/watch?v=UGvc-qujB-o>

Simulated Annealling

Cadernos de IA - 3_2 Têmpera Simulada

https://www.youtube.com/watch?v=S3_yOfVO0mo

Greedy Algorithms in Python: Optimal Task Assignment

<https://www.youtube.com/watch?v=QvSIAB27Vdk>

11.10. Python - CPLEX

<https://www.ibm.com/docs/en/icos/20.1.0?topic=cplex-python-reference-manual>

<https://community.ibm.com/community/user/datascience/blogs/xavier-nodet1/2020/07/09/cplex-free-for-students?CommunityKey=ab7de0fd-6f43-47a9-8261-33578a231bb7&tab=>

IBM SkillsBuild Software Downloads

<https://www.ibm.com/academic/home>

CPLEX currently only supports the API integration in Python 3.7 and 3.8 versions.

No terminal em MAC:

```
> python -m pip install --upgrade pip setuptools wheel
> pip install cplex-12.9.0.0-cp37-cp37m-macosx_10_6_x86_64.whl
```

No terminal em Windows (modo administrador) com a versão python 3.8.10:

<https://www.python.org/ftp/python/3.8.10/python-3.8.10-amd64.exe>

```
> pip install cplex
> pip install docplex
```

Correr o exemplo:

```
from docplex.mp.model import Model          # importar o módulo cplex
mdl = Model('Modelo')

x1 = mdl.continuous_var(name='x1')          # variáveis de decisão contínuas
x2 = mdl.continuous_var(name='x2')

mdl.add_constraint(x1 + x2 <= 80)             # restrições
mdl.add_constraint(2*x1 + x2 <= 100)
mdl.add_constraint(x1 <= 40)

mdl.maximize(27*x1 + 21*x2 - (10*x1 + 9*x2) - (14*x1 + 10*x2))

print(mdl.export_to_string())               # cria um ficheiro

solution = mdl.solve(log_output=True)
solution.display()
```

Imprime o resultado:

```
runfile('/Users/jpsantos/Documents/aulas/2019_2020SFP/Python/teste.py',
wdir='/Users/jpsantos/Documents/aulas/2019_2020SFP/Python')
\ This file has been generated by DOcplex
\ ENCODING=ISO-8859-1
\ Problem name: Modelo

Maximize
obj: 3 x1 + 2 x2
Subject To
c1: x1 + x2 <= 80
c2: 2 x1 + x2 <= 100
```

c3: x1 <= 40

Bounds

End

CPXPARAM_Read_DataCheck 1

Tried aggregator 1 time.

LP Presolve eliminated 1 rows and 0 columns.

Reduced LP has 2 rows, 2 columns, and 4 nonzeros.

Presolve time = 0.00 sec. (0.00 ticks)

Iteration log . . .

Iteration: 1 Dual infeasibility = 0.000000

Iteration: 2 Dual objective = 180.000000

solution for: Modelo

objective: 180.000

x1 = 20.000

x2 = 60.000

<http://ibmdecisionoptimization.github.io/docplex-doc/mp/docplex.mp.model.html>

11.11. Simulated Annealing – Sequenciamento

Para criar uma meta-heurística de sequenciamento, são necessários os seguintes passos:

- Criação de uma solução inicial – Pode ser gerada de forma aleatória ou através de um algoritmo guloso “greedy algorithm”.
- Cálculo do custo da solução inicial
- Para cada passo definido fazer:
 - o Calcular a temperatura
 - o Criar uma perturbação à solução inicial
 - o Calcular o novo custo
 - o Calcular probabilidade de aceitação
 - o Decidir a aceitação da nova solução

Em seguida são mostrados alguns métodos necessário à criação do algoritmo. Por questões de organização e boas práticas de programação, os métodos devem ser definidos num ficheiro .py e testados em outro ficheiro.

Iremos designar de SA.py ao ficheiro que contém os módulos e de teste.py ao ficheiro onde serão testados os métodos.

O primeiro passo, é importar os dados de sequenciamento de um ficheiro excel, para o ficheiro de teste:

```
import pandas as pd
import numpy as np
import SA as sa

filename = "exercicio.xlsx"
sheet_name = "Producao"
df = pd.read_excel(filename)
data = np.array(df)
Tasks = data[:, 0]
Time_to_process = data[:, 1]
Time_to_deliver = data[:, 2]
```

Quanto ao ficheiro SA.py, devem-se importar-se os seguintes módulos do numpy

```
import numpy as np
import numpy.random as rn
```

O seguinte método permite criar uma solução inicial aleatoriamente, a partir dos dados do ficheiro excel fornecido.

```
def random_start(x):
    """ Random sequence """
    y = x.copy()
    np.random.shuffle(y)
    return y
```

O seguinte método calcula o custo de cada solução, que neste caso consiste no cálculo do número de tarefas em atraso.

```
def cost_function(data):
    """ Calculates the number of delayed tasks """
    tasks = data[:, 0]
    time_to_process = data[:, 1]
    time_to_deliver = data[:, 2]
    time = 0
```


11.12. Algoritmos Genéticos – Sequenciamento

Os algoritmos genéticos, são outra meta-heurística que podemos utilizar em problemas de sequenciamento. Os passos seguintes são:

- Criação de uma população inicial (conjunto de possíveis soluções)– Pode ser gerada de forma aleatória ou através de um algoritmo guloso “greedy algorithm”.
- Para cada geração fazer:
 - o Cálculo do custo de cada elemento da população
 - o Selecionar os melhores indivíduos da população
 - o Crossover das melhores soluções
 - o Criação de uma nova população

Em seguida são mostrados alguns métodos necessários à criação do algoritmo. Por questões de organização e boas práticas de programação, os métodos devem ser definidos num ficheiro .py e testados em outro ficheiro.

Iremos designar de SA.py ao ficheiro que contém os módulos e de teste.py ao ficheiro onde serão testados os métodos.

O primeiro passo, é importar os dados de sequenciamento de um ficheiro excel, para o ficheiro de teste:

Para fazer e testar o algoritmo genético, sugere-se manter a mesma filosofia anterior, isto é, criar um ficheiro chamado “GA_sequence.py”, onde devem ser definidos todos os métodos necessários. O teste pode ser feito no ficheiro de testes criado anteriormente.

Relativamente ao ficheiro “GA_sequence.py”, serão fornecidos os métodos necessários à elaboração do algoritmo. O seguinte permite a criação de uma população aleatória, com um determinado tamanho.

```
def create_population(data, pop_size):
    data_copy = data.copy()
    population = []
    for i in range(pop_size):
        numpy.random.shuffle(data_copy)
        element = data_copy.copy()
        population.append(element)
    return population
```

O método a seguir permite calcular o custo de uma solução:

```
def cost_function(data):
    """ Calculating the number of delayed tasks """
    tasks = data[:, 0]
    time_to_process = data[:, 1]
    time_to_deliver = data[:, 2]
    time = 0
    delays = 0
    for i in range(len(tasks)):
        time = time + time_to_process[i]
        if time > time_to_deliver[i]:
            delays = delays + 1
    return delays
```

Enquanto o seguinte permite efetuar essa operação para uma população inteira:

```
def cal_pop_fitness(pop):
    """Calculating fitness vector for the entire population"""
    fitness = []
    for n in range(len(pop)):
        result = cost_function(pop[n])
        fitness.append(result)
    return fitness
```

O método seguinte faz a seleção dos indivíduos mais aptos de uma população:

[illegible]

O seguinte método altera as melhores soluções da população com vista a melhorá-las

```
def crossover(parents, offspring_size):
    """ The point at which crossover takes place between two parents. Usually, it is at the center. """
    parents_copy = parents.copy()
    offspring=[]
    i = 0
    for n in range(offspring_size):
        index_1 = numpy.random.randint(0, len(parents[0]))
        index_2 = numpy.random.randint(0, len(parents[0]))
        parent = parents_copy[i].copy()
        parent[[index_1, index_2]] = parent[[index_2, index_1]]
        i = i + 1
    if i >= len(parents_copy):
        i = 0
    offspring.append(parent)
    return offspring
```

O ficheiro de teste deve ter pelo menos as seguintes linhas:

```
import pandas as pd
import numpy as np
import simaneal_sequence as sa
import GA_sequence as GA

filename = "exercicio.xlsx"
sheet_name = "Producao"
df = pd.read_excel(filename)
```



```
data = np.array(df)
Tasks = data[:, 0]
Time_to_process = data[:, 1]
Time_to_deliver = data[:, 2]
pop_size = 10
num_generations = 1000
num_parents_mating = 3
```

Crie um algoritmo genético para otimizar o sequenciamento das tarefas já fornecidas.

11.13. RESTfull web service - Flask

Flask to create a RESTfull web service

<https://www.youtube.com/watch?v=s-i6nzXQF3g>

Criar/ensinar um modelo, usando um “Iris data set” com 4 atributos/variáveis: sepal length, sepal width, petal length, petal width e 3 classes de flores (para predizer): Iris-Setosa, Iris-versicolour, Irisverginica. existem por isso três classes , 3 Classificadores.

> pip install flask flask-material

```
iris = datasets.load_iris()
x= iris.data
y=iris.target
x_train, x_test, y_train, y_test = train_test_split(x, y)
rfc= RandomForestClassifier(n_estimator=100, n_jobs=2)
rfc.fit(x_train, y_train)
```

pickle: serialização em Python, consiste em converter um objecto, serializá-lo e neste caso guardar em disco. Por exemplo, depois de ensinar um modelo, pode-se guardar em disco, deixando de ser necessário ensinar o modelo de todas as vezes que ...

O objecto RainForestClassifier RFC pode ser guardado em disco:

```
pickle.dump(rfc, open(iris_rfc.pkl, "wb"))
```

O objecto pode voltar a ser criado a partir do ficheiro em disco:

```
my_random_forest = pickle.load(open(iris_rfc.pkl, "rb"))
```

---- Ficheiro em python “flask_demo.py”

```
import numpy as np
from flask import Flask, abort, jsonify, request
import pickle as pickle

my_random_forest = pickle.load(iris_rfc.pkl, "rb")

app= Flask(__name__)

@app.route('/api', methods=['POST'])
def make_predict():
    data = request.get_json(force=True)
    predict_request= [ data['sl'], data['sw'], data['pl'], data['pw'] ]
    predict_request = np.array(predict_request)
    y_hat = my_random_forest.predict(predict_request)
    output= [y_hat[0]]
    return jsonify(results=output)

if __name__ == '__main__':
    app.run(port=9000, debug=True)
```

executar o ficheiro/programa python, que implementa o REST API

```
> python flask_demo.py
running in 127.0.0.1 : 9000
```

A partir do cliente HTTP/TCP/IP (ex. browser):

```
url = http://localhost:9000/api  
data = json.dumps( { 'sl':5.84, 'sw':3.0, 'pl':3.75, 'pw':1.1 } )  
r = requests.post(url, data)
```

11.14. Python - SIMANNEAL

<https://pypi.org/project/simanneal/#files>
pip install simanneal

11.15. Python - Genetic

<https://pypi.org/project/sklearn-genetic/>
pip install sklearn-genetic

Exemplo:

```
from __future__ import print_function
import numpy as np
from sklearn import datasets, linear_model

from genetic_selection import GeneticSelectionCV

def main():
    iris = datasets.load_iris()

    # Some noisy data not correlated
    E = np.random.uniform(0, 0.1, size=(len(iris.data), 20))

    X = np.hstack((iris.data, E))
    y = iris.target

    estimator = linear_model.LogisticRegression(solver="liblinear", multi_class="ovr")

    selector = GeneticSelectionCV(estimator,
                                cv=5,
                                verbose=1,
                                scoring="accuracy",
                                max_features=5,
                                n_population=50,
                                crossover_proba=0.5,
                                mutation_proba=0.2,
                                n_generations=40,
                                crossover_independent_proba=0.5,
                                mutation_independent_proba=0.05,
                                tournament_size=3,
                                n_gen_no_change=10,
                                caching=True,
                                n_jobs=-1)
    selector = selector.fit(X, y)

    print(selector.support_)

if __name__ == "__main__":
    main()
```

estimator: ? número de dados/casos para ensinar ...?
n_jobs: ? número de classes para predizer ? 0,1 e 2

Machine Learning with Python - from Model to Web-Service by Aleksei Tiulpin

<https://www.youtube.com/watch?v=9sFUIR-CS7Y>

SciKit-learn:

- clf= randomForestClassifier (n_estimator= 100 arvores)
- clf.fit (classifier atributos m2 de cada casa)
- clf.predict

Caso de estudo: reconhecimento de dígitos MNIST

Pretende-se criar um WebService com as bibliotecas Flask e sic-kit learn. Desenhar um dígito no browser e ser capaz de reconhecer o dígito graças ao código em Backend

DataSet = 60 000 exemplos

test examples = 10 000

pictures = 28 x 28 pixel

11.16. Python - Pickle

Este módulo permite criar ficheiros contendo dados criados em python e ler dados de um ficheiro criado a partir desta biblioteca

O pickle pode ser instalado a partir do pip através do comando, na linha de comandos do pycharm
> pip install pickle

Para ser utilizada, deve-se importar, através: “import pickle” no cabeçalho do ficheiro .py.

Na figura abaixo, estão dois métodos, um para escrever dados num ficheiro cujo nome deve ser especificado e outro para ler dados de um ficheiro que foi previamente gravado com o pickle.

É de notar que os ficheiros .txt não são legíveis quando abertos num editor de texto, como o notepad ++, por exemplo, uma vez que os dados são encriptados antes de serem guardados em ficheiro.

```
def read_txt(nome_ficheiro):  
    arq = open(nome_ficheiro, 'rb') # abrir o arquivo para leitura - o "b" significa que o arquivo é binário  
    acidentes_way = pickle.load(arq) # Ler a stream a partir do arquivo e reconstroi o objeto original.  
    arq.close() # fechar o arquivo  
    return acidentes_way  
  
def write_txt(nome_ficheiro, dados):  
    arq = open(nome_ficheiro, 'wb') # open file- "b" means binary  
    pickle.dump(dados, arq) # Save stream object in file  
    arq.close() # close the file
```

https://www.w3schools.com/python/python_file_handling.asp

11.17. Python Mongo DB

https://www.w3schools.com/python/python_mongodb_getstarted.asp

11.18. Python - Keras Tensor FlowLibrary

Convolutional Networks

11.19. Python Ficheiros de texto

https://www.w3schools.com/python/python_file_handling.asp

Solution of Job Shop Scheduling (JSS) Problem/ N Jobs on M Machines Problem using GA

<https://www.youtube.com/watch?v=5OgBQCMUUqM>

AI for Supply Chain (ok)

<https://www.youtube.com/watch?v=vwor9Fva1V4>

- 3 centros de distribuição
- 5
- 4 centros de retalho (venda ao público)

Simular uma tempestade. Trajectos:

Salt lake to Boston

Salt lake to Raleigh

O objectivo é aumentar o lucro, o retorno do investimento.

Pretende-se adaptar as rotas de aprovisionamento, as datas de envio/transporte e entrega, pedidos dos clientes e em função das tempestades

Optimizing Delivery Routes - Intro to Theoretical Computer Science

<https://www.youtube.com/watch?v=MJ76MeuckWM>

(apresenta bem o problema do caixeiro viajante, mas não tem código ...)

Job Sequencing Problem (Greedy Algorithm) | GeeksforGeeks

<https://www.youtube.com/watch?v=R6Skj4bT1HE>

(ok- explica o problema e com código)

3.2 Job Sequencing with Deadlines - Greedy Method

<https://www.youtube.com/watch?v=zPtI8q9gvX8>

(explica mas não tem código)

Iniciando com Orange

<https://www.youtube.com/watch?v=HXjnDIgGDuI&list=PLmNPvQr9Tf-ZSDLwOzxpY-HrE0yv-8Fy>

01: bem-vindos ao Orange

02: workflows de dados (fluxos de trabalho de dados)

03: Widgets e Canais

04: Loading your data

05: Hierarchical clustering

06: Fazendo previsões

(predizer se é um vegetal, ou fruto)

- classification se é Vegetal ou fruto: Vitamina A, Cálcio, magnésio, potássio, nome ..
- classification tree (classificar os dados com arvores de decisão)
- regressão linear
 - construir modelos de predição/classificação
 - e usá-los nos conjuntos de dados adquiridos

07: Avaliação e Pontuação do Modelo

<https://www.youtube.com/watch?v=pYXOF0jziGM>

(que modelo se comporta melhor ?)

09: Principal component analysis

Operation Scheduling Using Genetic Algorithm in Python

<https://www.youtube.com/watch?v=e84aLKGWtW4>

Classification in Orange (CS2401)

<https://www.youtube.com/watch?v=G3W2Jc7Wtfw>

Operation Scheduling Using **Genetic Algorithm in Python**

<https://www.youtube.com/watch?v=e84aLKGWtW4>

How to Deploy a Tensorflow **Model to Production**

https://www.youtube.com/watch?v=T_afaArR0E8

Applications of Machine Learning in the **Supply Chain**

<https://www.youtube.com/watch?v=pzzFvhJ6-LI>

PyTorch Developer Conference 2018: Production & Research Sessions

<https://www.youtube.com/watch?v=mGycZTc1SxY>

Deploying **Predictive** Models in Python and R

<https://www.youtube.com/watch?v=P75D9FgTP0Q>

Scaling Machine Learning jobs with **Kubernetes** - Tarek Mehrez, Carsten Lygteskov Hansen

<https://www.youtube.com/watch?v=fYGulideQkw>

Create A Neural Network That Classifies Diabetes Risk In 15 Lines of Python

<https://www.youtube.com/watch?v=T91fsaG2L0s>

How to deploy machine learning models into production

<https://www.youtube.com/watch?v=-UYyyeYJAoQ>

- <https://www.ibm.com/cloud/cloud-foundry>
- <https://kubernetes.io>
- <https://www.ibm.com/cloud/machine-learning>

How to Deploy Keras Models to Production

<https://www.youtube.com/watch?v=f6Bf3gl4hWY>

Deep-Learning-in-Production meetup Jan, 2019, Mobile training, Deep Learning for NLP at HubSpot

<https://www.youtube.com/watch?v=0MGpyYovUwM>

Spark: Interactive To Production

https://www.youtube.com/watch?v=jJqV_6o3plM

What is PRODUCTION ENGINEERING? What does PRODUCTION ENGINEERING mean?

<https://www.youtube.com/watch?v=2mAeggAH28E>

Machine Learning Solutions For The Manufacturing Industry | Industry 4.0 Solutions
<https://www.youtube.com/watch?v=IQvH6NZOJNo>

Machine Shop Scheduling for High Mix Low Volume
<https://www.youtube.com/watch?v=57u9D0vQjjA>

Webinar: 6 Ways machine learning is revolutionizing Manufacturing
<https://www.youtube.com/watch?v=rSDXiJU43Ag>

11.20. Python SciKit-Learn

SciKit-Learn (biblioteca de algoritmos: árvores de decisão, redes neurais,)

Em python existem algoritmos para machine learning de dois tipos Supervised and Unsupervised:

Supervised Learning:

- Decision trees
- Support Vector Machines
- Naïve Bayes
- k-nearest neighbor
- Linear regression

Unsupervised Learning (Clustering, Association):

- K-means clustering
- Hierarchical clustering

Python:

- 1 Instalar o módulo SciPy.
- 2 Adquirir os dados.
- 3 Tratar os dados.
- 4 Visualizar/analisar os Dados.
5. Treinar os algoritmo
- 5 Avaliar o algoritmo.
- 6 Fazendo algumas previsões.

Python tutorial for beginners (Mosh)

https://www.youtube.com/watch?v=_uQrJ0TkZlc&t=17731s

(cria 3 projetos em Python)

- 1 - explica com instalar o Python no computador, download <http://python.org> and install
- 2 - usar um editor para escrever o código, por exemplo pycharm IDE. Download pycharm and install
- 3- python para web app, desktop app , AI ?
- 4- apresenta : variáveis, if, for, funções, parâmetros, ...
- 4- instalar libraries

> *pip install openpyxl //para trabalhar com excel instala a biblioteca no directório "python 3.7\site-packages"*

- 5- programa em python para processar dezenas de folhas de cálculo excel, alterar uma fórmula ou um gráfico.

```
import openpyxl as xl
from openpyxl.chart import BarChart, Reference
```

```
wb= xl.load_workbook('ficheiro.xlsx')
sheet= wb['sheet1']
cell= sheet['a1']
cell= sheet.cell(1,1)
print(cell.value)
print(sheet.max_row)
```

```
for row in range(2, sheet.max_row+1):
    cell=sheet.cell(row,3)
    print(cell.value)
```

```
wb.save('ficheiro2.xlsx')
```

```
values = Reference(sheet, min_row=2, max_row=sheet.max_row, min_col=4, max_col=4)
chart = BarChart()
chart.add_data(values)
sheet.add_chart(chart, 'e2')
```

6- machine learning (4h12min)

Numpy (multidimensional array)

Pandas (data analysis, data frame de duas dimensões, linhas e colunas)

Mathplotlib (para criar gráficos de duas dimensões)

SciKit-Learn (biblioteca de algoritmos: árvores de decisão, redes neuronais,)

1- Instalar o Django

2- usar o Jupyter para editar e executar os programas em python

3- importar um conjunto de dados (data set) www.kaggle.com video game sales (vgsales.csv)

data

```
import pandas as pd
```

```
df = pd.read_csv(vgsales.csv)
```

```
df //imprime toda a tabela
```

```
df.shape // imprime a dimensão da tabela 16598,11)
```

4- machine learning

**** Import data**

```
import pandas as pd
```

```
music_data = pd.read_csv(music.csv)
```

**** Clean data (remover dados duplicados, nulos, ..)**

**** Split data (dividir as colunas, input e output)**

id, age, gender (masculino-0/feminino-1), genre (género de musica)

0, 20, 1, hiphop

1, 23, 1, hiphop

2, 25, 1, Hiphop

3, 26, 1, Jazz

4, 29, 1, Jazz

9, 20, 0, Dance

10, 21, 0, Dance

11, 25, 0, Dance

12, 26, 0, Acoustic

13, 27, 0, Acoustic

```
x = music_data.drop(columns=['genre']) // remove a coluna 'genre', não altera o ficheiro original
```

```
// X passa a conter id, age, gender
```

```
y = music_data['genre'] // y passa a conter a tabela id, genre (género musical)
```

**** Create model**

**** Train model**

**** Make prediction**

```
from sklearn.tree import DecisionTreeClassifier
```

```
model = DecisionTreeClassifier() // create model
```

```
model.fit(x,y) // Train model
```

```
predictions = model.predict([ [21,1], [20,0] ]) // pergunta qual música preferida de uma mulher de 21 anos  
// e de um homem de 20 anos.
```

```
predictions // responde 'hipHop', 'Dance'
```

**** Evaluate and improve**

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.2) // a partir dos dois arrays 'x' e 'y' cria 4 arrays  
model = DecisionTreeClassifier  
model.fit(x_train, y_train)  
predictions = model.predict(x_test)  
score = accuracy_score(y_test, prediction)
```

4:50m Decision tree, graphs ...

4:58 Criar um web application/ Web site

1- instalar Django (tem módulos para tratar de HTTP request, url, sessions, cookies)

Anaconda Navigator : jupyterlab, notebook, spyder (Python), GlueViz, Orange, RStudio, VS Code.