Análise de dados e computação científica em Python

Exercícios

1. Se ainda não o fez, instale um Ambiente Integrado de Desenvolvimento (IDE) para desenvolvimento de software e programação em Python.

Se ainda tem dificuldades, talvez prefira iniciar com este IDE simples: https://thonny.org/

Uma das características da linguagem de programação Python é que os elementos e construtores da linguagem podem ser interpretados como objetos, incluindo as funções.

2. O passo seguinte é criar e manipular ficheiros de dados e respetivos caminhos (paths).

Experimente criar um ficheiro de texto e gravá-lo em diversas directorias.

De seguida leia esse mesmo ficheiro e importe a informação para variáveis.

3. Geração aleatória de dados. A biblioteca (módulo base do ambiente Python) **random** permite a geração aleatória de dados.

Experimente criar um dicionário com valores numéricos aleatórios entre 0 e 1.

```
import random
dados = { i : random.random () for i in range (10) }
print (dados)
```

De seguida repita o exercício, utilizando o módulo **numpy**.

```
from numpy.random import randn
dados2 = { i : randn () for i in range (10) }
print (dados2)
```

4. A biblioteca **numpy** – **Num**erical **Py**thon – é um módulo essencial para o desenvolvimento de análise de dados numéricos e computação científica. As funções associadas a este módulo permitem o processamento de *arrays* com uma sintaxe mais simples, numa computação e programação vectorizada.

Crie um *array* unidimensional (ou vector) e um *array* bidimensional (matriz).

```
import numpy as npy
v=npy.array ([4,6,3,32,99,47,100,25])
v.ndim
len(v)
M= npy.array ( [ [5,16,23,12],[49,7,100,15] ] )
M.ndim
len(M)
M.shape
```

Experimente as funções **zeros**, **ones**, **eye**, **diag** que permitem criar de forma semiautomática diversos *arrays* úteis.

Guarde algumas destas matrizes em ficheiros de dados.

5. Crie uma matriz, de 10 linhas e 5 colunas, com números inteiros aleatórios entre 0 e 1000.

Guarde essa matriz num ficheiro de dados, com as colunas separadas por TAB.

Guarde a mesma matriz num ficheiro de dados, com as colunas separadas por vírgulas.

6. Implemente um programa que utilize um *array* **numpy** para determinar os números primos num determinado intervalo de valores inteiros, por exemplo o intervalo [0,90].

7. Implemente um programa para determinar os números divisores por 3 de um *array* **numpy** com 20 elementos criado aleatoriamente num determinado intervalo [0,70].

8. A representação e manipulação de estruturas de dados é a principal componente da biblioteca **pandas**. Implemente um programa que permita importar os dados de precipitação total anual (mm) do ficheiro **PrecipitacaoPT.csv**.

De seguida calcule algumas estatísticas básicas para cada cidade (valores mínimo, máximo, médio,etc.).

NOTA: É também facultado o ficheiro **PrecipitaçãoPT.xlsx**, para que possa consultar a metainformação.

9. A visualização gráfica dos dados é efetuada através do módulo **matplotlib**, que permite a utilização de funções de visualização de dados em duas dimensões (2D). Comece por fazer e personalizar figuras simples como seja a representação das funções trigonométricas seno e cosseno (exemplo de gráfico simples de linhas, ***plot***).

```
import numpy as npy
import matplotlib.pyplot as plt
x=npy.linspace(-npy.pi*2, npy.pi*2,256)
y1=npy.sin(x)
y2=npy.cos(x)
plt.plot(x,y1,color='red',linestyle='-',label='seno')
plt.plot(x,y2,color='blue',linestyle=':',label='cosseno')
plt.legend(loc='upper right')
plt.show()
```

10. Utilizando os dados de precipitação total anual (mm) do ficheiro **PrecipitacaoPT.csv**, faça alguns gráficos básicos, de linhas (função ***plot***) e de pontos (função ***scatter***). Represente, por exemplo:

a) A evolução temporal da precipitação em Viana do Castelo (Figura 1 – série temporal)
b) A precipitação total em Viana do Castelo *versus* Porto (Figura 2 –diagrama de dispersão).

Personalize as figuras (título, eixos, legendas, etc).

Grave as figuras em diversos formatos, adequados para inserir num relatório.