Programació per a *Data Science*

Unitat 4: Llibreries científiques en Python

Aquesta Unitat està formada per 4 mòduls: Matplotlib, Numpy, pandas i SciPy. A continuació, us proposem una serie d'exercicis a realitzar per cada un d'aquests mòduls.

Exercicis per a practicar

Els següents 4 exercicis no puntuen per a la PAC, però us recomanem que els intenteu resoldre abans de passar als exercicis propis de la PAC. També trobareu les solucions a aquests exercicis al final del Notebook.

Exercici 1

Calculeu la norma i el determinant de la següent matriu: `[[1, 0], [2, -1]]`.

In [ ]:

```
# Resposta
```

Exercici 2

Avalueu les funcions arcsinus i arccosinus a l'interval [0,1] i amb pas (resolució) de 0.1 i guardeu-les en dos *arrays*.

In [ ]:

```
# Resposta
```

Exercici 3

Representeu a una sola gràfica les funcions arcsinus i arccosinus a l'interval [-pi/4, pi/4]. Afegiu la llegenda corresponent a cada gràfic.

In [ ]:

```
# Resposta
```

Exercici 4

Genereu una llista de 100 valors enters aleatoris de 0-9. Realitzeu els següents càlculs utilitzant mètodes de *numpy*:

- Mitjana i desviació estàndard dels valors de la llista
- Valor màxim i mínim
- Sumeu tots els valors de la llista
- Aconseguiu una llista de valors únics

In [ ]:

```
# Resposta
```

Exercicis per a la PAC

A continuació, trobareu els exercicis que heu de completar en aquesta PAC i que formen part de l'avaluació d'aquesta unitat.

NumPy - Exercicis

Exercici 1

Ordeneu la matriu bidimensional `[[5,1,7], [0,7,4], [7,23,1]]` per files utilitzant com a algorisme d'ordenació el Merge sort (https://en.wikipedia.org/wiki/Merge_sort). **(0.5 punts)**

Hint: No es necessari que implementeu l'algorisme d'ordenació a mà, NumPy conté mètodes per a realitzar diferents tipus d'ordenació sobre diferents estructures de dades.

In [ ]:

```
# Resposta
```

Exercici 2

Definiu una funció que donades dues matrius, retorni el valor absolut de la multiplicació dels determinants d'ambdues, és a dir, donades A i B, la nostre funció retornarà $|det(A) * det(B)|$. **(0.5 punts)**

In [ ]:

```
# Resposta
```

Exercici 3

Creeu una matriu 10x10 que correspongui amb la matriu identitat (https://cat.wikipedia.org/wiki/Matriu_identitat) utilitzant generadors bàsics d'arrays. Creeu una matriu identitat 10x10 (aquest cop utilitzant generadors específics de matrius identitat) i comproveu que ambdues matrius són iguals **(0.5 punts)**

Consideracions:

- La primera matriu ha de crear-se utilitzant constructors bàsics d'arrays, com els presentats als Notebooks de teoria.
- La segona matriu ha de generar-se utilitzant el generador de matrius identitat de numpy.
- La comparació ha de retornar True si les matrius són iguals (un únic True), False en cas contrari.

In [ ]:

```
# Resposta
```

Exercici 4

Creeu una matriu de 2x6 on els valors de cada posició `(i, j)` corresponguin a `i^2+j` per a tot `i` parell, `i/j^2` per tot `i` senar. **(0.5 punts)**

In [ ]:

```
# Resposta
```

Exercici 5

Creeu dues matrius de mida 5x5 amb números reals aleatoris. Obteniu el resultat de multiplicar ambdues matrius utilitzant els dos mètodes de multiplicació de matrius vistos al Notebook de teoria. Quina és la diferència entre els resultats? Creeu ara dos matrius de mida 4x5 i 5x5 respectivament, repetiu la operació. Descriviu quin dels mètodes de multiplicació podeu aplicar i perquè. **(0.5 punts)**

In [ ]:

```
# Resposta
```

Matplotlib - Exercicis

Exercici 1

Representeu en un únic gràfic les funcions f1 i f2 definides mes avall, avaluades en l'interval [0, 7] i amb pas (resolució) 0.1. Definiu una llegenda que inclogui el significat de cada funció (2x, x^2) i posicioneu-la de tal forma que no tapi els gràfics. **(1 punt)**

In [ ]:

```
import numpy as np

def f1(x):
    return np.power(x, 2)

def f2(x):
    return np.power(2,x)
```

In [ ]:

```
# Resposta
```

Exercici 2

L'arxiu *us_births.csv* conté dades dels naixements a Estats Units durant els anys 1994-2003. Carregueu l'arxiu representeu de forma gràfica els naixements agrupats por mes i any. Utilitzeu un gràfic 2D de tipus *scatter*. **(1.5 punts)**

Consideracions:

- La figura ha de contenir 10 gràfics (1 per any)
- El valor de cada punt ha de correspondre amb el total de naixements per any i mes.
- Incloeu una llegenda que contingui els anys. Podeu diferenciar cada gràfic como millor us sembli (diferents símbols / colors / combinació de les dues)

In [ ]:

```
# Resposta
```

pandas - Exercicis

Exercici 1

Carregueu les dades de l'arxiu *got.csv* a un *dataframe*. Aquest conjunt de dades recull informació de la Guerra dels Cinc Reis (https://awoiaf.westeros.org/index.php/War_of_the_Five_Kings) de les novel·les de Cançó de gel i de foc (https://ca.wikipedia.org/wiki/Can%C3%A7%C3%B3_de_gel_i_de_foc) de George R.R Martin.

Mostreu el número de files del *dataframe* i les etiquetes dels eixos. **(0.5 punts)**

In [ ]:

```
# Resposta
```

Exercici 2

Agrupeu les dades carregades a l'exercici 1 pel principal bàndol atacant (*attacker_1*). Per cada posició, mostreu el número de batalles i el resultat de la batalla (el resultat es troba al camp *attacker_outcome*). **(0.5 punts)**

In [ ]:

```
# Resposta
```

Exercici 3

Mostreu les dades de las batalles on el número de participants supera els 15000 "homes" (comptant els dos bàndols: *attacker_size* i *defender_size*), el resultat hagi estat desfavorable per l'atacant, i la batalla hagi estat a l'hivern (*summer=0*). **(1 punt)**

In [ ]:

```
# Resposta
```

Exercici 4

Compteu quants llocs apareixen més d'un cop al *dataframe* (camp *location*), utilitzant les funcions de la llibreria *pandas*. Existeix algun lloc on hi hagi hagut més d'una batalla? Comproveu quins bàndols hi estaven implicats. **(0.5 punts)**

In [ ]:

```
# Resposta
```

Exercici 5

Afegiu una nova columna al *dataframe* amb un valor booleà corresponent a una predicció bàsica sobre si en una certa batalla el bàndol atacant serà el guanyador. Definireu el valor com a *True* si el bàndol atacant és major al defensor, i como a *False* en cas contrari. Definiu como a *NaN* aquells casos on no hi hagi número de tropes en cap bàndol. A quins casos, identificats pel número de batalla (*battle_number*), la nostra predicció coincideix amb el resultat real? **(1 punt)**

In [ ]:

```
# Resposta
```

SciPy - Exercicis

Hint: Podeu utilitzar el mòdul integrate (https://docs.scipy.org/doc/scipy/reference/integrate.html#module-scipy.integrate) de SciPy per a resoldre els diferents tipus d'integrals.

Exercici 1

Resoleu la següent integral definida: **(0.5 punts)**

$$\int_0^5 x^2 + 4x + 3dx$$

In [1]:

```
# Resposta
```

Exercici 2

Resoleu la següent integral doble definida: **(1 punt)**

$$\int_{x=0}^{\pi}\int_{y=\pi}^{2\pi} 3x\sin(y^2)+6y\sin(x^2)dydx$$

In [ ]:

```
# Resposta
```

Solucions als exercicis per a practicar

Exercici 1

Calculeu la norma i el determinant de la següent matriu: `[[1, 0], [2, -1]]`.

In [2]:

```
import numpy as np

# Creem la matriu com a array bidimensional
m = np.array([[1, 0], [2, -1]])

# Calculem la normal i el determinant utilitzant el mòdul linalg
norm = np.linalg.norm(m)
det = np.linalg.det(m)

print "La matriu m és:", m
print "La norma de m és: ", norm
print "El determinant de m és:", det
```

```
La matriu m és: [[ 1  0]
 [ 2 -1]]
La norma de m és:  2.449489742783178
El determinant de m és: -1.0
```

Exercici 2

Avalueu les funcions arcsinus i arccosinus a l'interval [0,1] i amb pas (resolució) de 0.1 i guardeu-les en dos *arrays*.

In [4]:

```
import numpy as np

# Generem un vector amb valors entre 0 i 1 i amb pas 0.1. Utilitzem la funció ar
ange,
# que funciona de forma anàloga a la funció range vista en unitats anteriors.
x = np.arange(0., 1.1, 0.1)

# Calculem el valor del arcsinus i del arccosinus per a cada valor de x utilitza
nt les
# funcions arcsin i arccos de numpy
arcsin = np.arcsin(x)
arccos = np.arccos(x)

print "Arcsinus de x per a x entre 0 i 1, amb pas 0.1:"
print arcsin
print "Arccosinus de x per a x entre 0 i 1, amb pas 0.1:"
print arccos
```

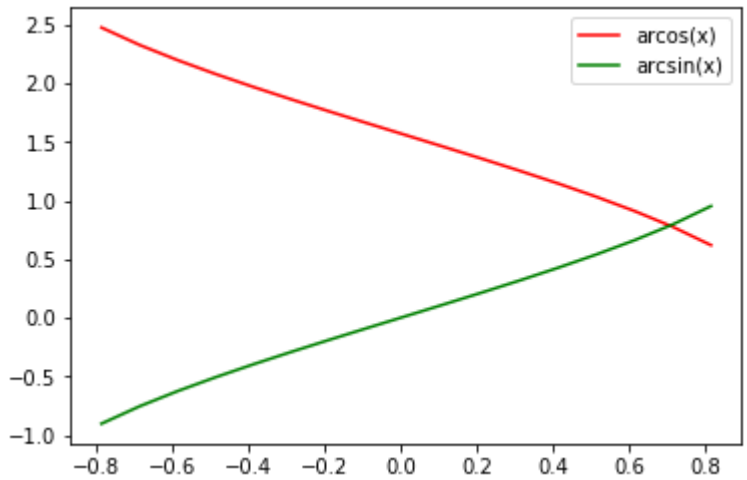
```
Arcsinus de x per a x entre 0 i 1, amb pas 0.1:
[0.         0.10016742 0.20135792 0.30469265 0.41151685 0.52359878
 0.64350111 0.7753975  0.92729522 1.11976951 1.57079633]
Arccosinus de x per a x entre 0 i 1, amb pas 0.1:
[1.57079633 1.47062891 1.36943841 1.26610367 1.15927948 1.04719755
 0.92729522 0.79539883 0.64350111 0.45102681 0.        ]
```

Exercici 3

Representeu a una sola gràfica les funcions arcsinus i arccosinus a l'interval [-pi/4, pi/4]. Afegiu la llegenda corresponent a cada gràfic.

In [5]:

```
%matplotlib inline

import matplotlib
import numpy as np
import matplotlib.pyplot as plt

# Comencem calculant els valors de l'interval -pi/4 a pi/4. No se'ns ha especifi
cat cap pas, pel que
# triarem un nosaltres mateixos. Per a donar una resolució suficientment alta, t
riarem un pas baix (quant
# més baix el pas, més elements a la llista i per tant, major resolució).
pas = 0.1
x = np.arange(-np.pi/4., np.pi/4. + pas, pas)

# A continuació representarem ambdues funcions al mateix gràfic
# Fixeu-vos que els paràmetres de la funció plot corresponen a:
# grafic_1_eix_x, grafic_1_eix_y, color_grafic_1, grafic_2_eix_x, grafic_2_eix_
y, color_grafic_2, ...
plt.plot(x, np.arccos(x), 'r', x, np.arcsin(x), 'g')

# Afegim la llegenda al gràfic
plt.legend(["arcos(x)", "arcsin(x)"])

# I el mostrem per pantalla
plt.show()
```



Exercici 4

Genereu una llista de 100 valors enters aleatoris de 0-9. Realitzeu els següents càlculs utilitzant mètodes de *numpy*:

- Mitjana i desviació estàndard dels valors de la llista
- Valor màxim i mínim
- Sumeu tots els valors de la llista
- Aconseguiu una llista de valors únics

In [8]:

```
from random import randint
import numpy as np

# Utilitzem la funció randint del mòdul random i list comprehensions per a gener
ar la llista, com ja havíem vist a
# unitats anteriors
rand_ints = [randint(0, 9) for _ in range(100)]

# Per a calcular la mitjana i la desviació estàndard utilitzem les funcions mean
y std (standard deviation)
#respectivament
m = np.mean(rand_ints)
std = np.std(rand_ints)

print "La nostra llista de valors aleatoris és:", rand_ints
print "La seva mitjana és %f, i la seva desviació estàndard és %f" %(m, std)

# Per als valors màxim i mínim, utilitzem max i min
max_val = np.max(rand_ints)
min_val = np.min(rand_ints)

print "Els seus valors màxim i mínim són, respectivament: %d i %d" %(max_val, mi
n_val)

# Per a sumar tots els elements de la llista, la funció sum
sum_values = np.sum(rand_ints)

print "El resultat de la suma de tots els valors de la llista és", sum_values

# I finalment per a aconseguir una llista de valors únics, la funció unique
unique_values = np.unique(rand_ints)

print "I la llista de valors únics és", unique_values
```

```
La nostra llista de valors aleatoris és: [0, 0, 9, 6, 4, 8, 3, 6, 8,
4, 4, 9, 7, 6, 8, 2, 7, 8, 3, 2, 8, 4, 4, 6, 7, 2, 6, 4, 8, 9, 2, 9,
6, 8, 9, 0, 0, 7, 4, 8, 3, 2, 1, 6, 5, 5, 7, 1, 5, 6, 3, 7, 1, 0, 2,
7, 2, 6, 8, 9, 3, 8, 6, 3, 8, 8, 5, 5, 1, 2, 3, 2, 0, 8, 8, 6, 2, 4,
3, 5, 7, 9, 0, 7, 7, 0, 3, 9, 9, 0, 1, 6, 4, 3, 9, 7, 5, 8, 7, 9]
La seva mitjana és 5.010000, i la seva desviació estàndard és 2.8548
03
Els seus valors màxim i mínim són, respectivament: 9 i 0
El resultat de la suma de tots els valors de la llista és 501
I la llista de valors únics és [0 1 2 3 4 5 6 7 8 9]
```