TP n°1: Introduction à python pour le calcul scientifique

♦ **Exercice 1** Dessiner un graphe donnant le temps d'exécution de la somme de deux matrices carrées en fonction de n la taille des matrices.
Précisions sur le travail demandé :

- n sera un entier compris entre 2 et 1 000 ;
- les matrices seront représentées par des objets de la classe `array` de `numpy`
- le graphe comportera deux courbes l'une donnera le temps d'exécution de la fonction `add` de `numpy` et l'autre celui de la somme réalisée par une fonction codée par vos soins.

Memo python/numpy

- Charger `numpy`

```
import numpy as np
```

- Récupérer la date et l'heure

```
from time import time
temps=time()
```

- Créer une matrice de taille n

```
M=np.random.rand(n,n)
```

- Tracer les courbes grâce à `matplotlib`

```
import matplotlib.pyplot as plt
cm = 1/2.54 # 1 pouce = 2.54 cm
# Créer une figure avec un axe
fig, ax = plt.subplots(figsize(12*cm,8*cm))
# n, y1 et y2 étant de même dimension

# Tracer une première courbe
ax.plot(n, y1, label='étiquette 1')
# Tracer une seconde courbe
ax.plot(n, y2, label='étiquette 2')

# Ajouter une légende pour les axes
ax.set_xlabel('étiquette abscisse')
ax.set_ylabel('étiquette ordonnée')
# Ajouter un titre et les étiquettes
ax.set_title("Graphique")
ax.legend()
```

♦ **Exercice 2** Soit $P = a_0 + a_1 X + a_2 X^2 + \cdots + a_{n-1} X^{n-1} + X^n$ un polynôme unitaire de $\mathbb{C}_n[X]$ l'ensemble des polynômes de degré inférieur ou égal à $n \in \mathbb{N}^*$ à coefficients dans $\mathbb{C}$. On associe à P la matrice suivante, appelée matrice compagnon du polynôme P :

$$C_p = \begin{pmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \ddots & \vdots & -a_2 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & 1 & -a_{n-1} \end{pmatrix}$$

Écrire une fonction `python` qui à un élément P de la classe `Polynomials` de `numpy` renvoie la matrice compagnon associée au polynôme normalisé de P sous forme d'un tableau `numpy`. La fonction ne devra comporter aucune boucle !

Memo python/numpy

- Charger la classe Polynomial

```
from numpy.polynomial import Polynomial as P
```

- Créer le polynôme $X^3 - 2X + 1$

```
p=P([1,-2,0,1])
```

- Récupérer les coefficients de p

```
p.coef
```

- Récupérer le degré de p

```
p.degree()
```

- Créer une matrice nulle de taille 5×7

```
np.zeros((5,7))
```

- Créer la matrice identité de taille 5

```
np.eye(5)
```

TP n°1: Introduction à python pour le calcul scientifique

♦ **Exercice 3** À partir des données FIE Fencing Womens Foil Data, représenter l'évolution des classements dans le top 16 mondial, des tireuses françaises dans la catégorie sénior entre 2003 et 2022.

Memo python/pandas

- Charger la classe pandas et récupérer les données

```
import pandas as pd
data=pd.read_csv('fichier.csv')
```

- afficher des informations sur une table

```
data.infos()
```

- Filtrer une table

```
data[(data.colonne_1>10) & (data.colonne_2.isin(['bleu','vert'])) ]
```

♦ Exercice 4

Le jeu de données «palmerpenguins» (1) est largement utilisé pour introduire les principaux concepts d'apprentissage machine (Artwork by @allison_horst).

1. Charger le jeu de données à l'aide de `seaborn`.
2. Décrire chaque colonne du jeu de données (nature des caractères, distributions, etc.)
3. Certaines colonnes sont-elles corrélées ?
4. Représenter – de plusieurs manières différentes – la répartition des trois espèces *Adelie*, *Gentoo*, *Chinstrap* en fonction de la longueur du bec, de la hauteur du bec et de la longueur des nageoires. Quels commentaires peut-on faire ?

Memo python/seaborn

- Charger la classe seaborn et récupérer les données

```
import seaborn as sns
pingouins=sns.load_dataset('penguins')
```

- Calculer la matrice de corrélation

```
data.corr()
```

- Afficher une carte de chaleur

```
palette=sns.diverging_palette(
        230, 20, as_cmap=True)
sns.heatmap(MatriceDeCorrelation,
         annot=True,cmap=palette)
```

- Explorer les données

```
#pairplot où l'on regroupe les éléments de
# la  colonne 'cat' par couleur
sns.pairplot(donnees,hue='cat',
       diag_kind='kde',kind='scatter')
#Boîtes à moustache
sns.boxplot(x=, y=, data=donnees)
#Coordonnées parallèles
from pandas.tools.plotting
import parallel_coordinates as pc
pc(donnees,'species')
```

- Homogénéiser les données

```
from sklearn.preprocessing import
     StandardScaler, MinMaxScaler
scaler=StandardScaler()
data[colonnesListe]=scaler
    .fit_transform(data[colonnesListe])
```

Références

1. A. M. Horst, A. P. Hill, K. B. Gorman, palmerpenguins : Palmer Archipelago (Antarctica) penguin data, (https://allisonhorst.github.io/palmerpenguins/).