المملكة المغربية
ⵜⴰⴳⵍⴷⵉⵜ ⵏ ⵍⵎⵖⵔⵉⴱ

وزارة التربية الوطنية
والتعليم الأولي والرياضة
ⵜⴰⵎⴰⵡⴰⵜ ⵏ ⵓⵙⴳⵎⵉ ⴰⵏⵎⵓⵙ
ⴰ ⵓⵙⵙⵏⵎⴰ ⴰⵎⴰⵡⴰⵙ ⴰ ⵜⵓⵍⵍⵓⵜ

الأقسام التحضيرية للمدارس العليا
تازة

# Rapport de stage

## d'habilitation à enseigner dans les

## Classes Préparatoires aux Grandes Écoles

Informatique

Youssef Daghouj

2024/2025

# Table

# I

# Remerciement

Si ce travail a pu voir le jour, c'est certainement grâce à plusieurs personnes, c'est la raison pour laquelle je profite de cette opportunité pour exprimer ma gratitude et mes sincères remerciements et ce particulièrement à :

Monsieur Brinsi Zoubir, le directeur du lycée *Acharif Alidrissi Taza*, et du centre des classes préparatoires aux grandes écoles, Monsieur Lazrak Mohammed le directeur d'études et Monsieur Elyadari Abdeljabar le surveillant général pour leur professionnalisme, aide et disponibilité tout au long de cette expérience professionnelle.

Monsieur Kanber Ahmed, le chargé d'inspection d'informatique, pour ses efforts, ses conseils ainsi que son professionnalisme.

Mon collègue professeur Abid Abdallah pour sa disponibilité, ses conseils et les échanges constructifs que nous avons eus autour des pratiques et des élèves.

Le corps professoral et administratif et toutes les personnes qui m'ont aidé, de près ou de loin, d'une manière ou d'une autre, à la réussite de ce stage.

# 2
# Introduction

Les Classes Préparatoires aux Grandes Écoles constituent un cycle post-baccalauréat de deux années scolaires et occupent une place à part dans le paysage de l'enseignement supérieur. Véritables tremplins vers les écoles d'ingénierie et de commerce, elles offrent une formation avec un rythme soutenu afin de consolider et approfondir les connaissances acquises au lycée tout en développant des méthodes de travail efficaces et une grande autonomie de réflexion.

Ce stage a été une excellente opportunité pour découvrir les enjeux et les spécificités de ce cycle, aussi bien sur le plan des connaissances que sur les méthodologies de l'enseignement.

Le présent rapport résume, pour chaque classe, les objectifs et le programme de cette année de stage.

# 3
# Déroulement

Durant ce stage, on a eu l'occasion d'enseigner cinq classes, dont trois en première année et deux en deuxième année, appartenant à deux filières différentes :

- Une classe MPSI;
- Deux classes en 1ère année ÉCS;
- Deux classes en 2ème année ÉCS.

## 3.1 MPSI

### 3.1.1 Structure

Une classe de 33 élèves répartie en deux groupes. L'enseignement de chaque groupe de classe se fait avec une fréquence de deux heures par semaine.

### 3.1.2 Objectif général

En première année, le programme d'informatique est élaboré pour doter les élèves des connaissances et des outils nécessaires afin de découvrir et d'exploiter la puissance de l'informatique dans la résolution des problèmes liés aux mathématiques, à la physique et aux sciences de l'ingénieur.

### 3.1.3 Contenu du programme

Afin d'optimiser l'apprentissage et de répondre de manière ciblée aux besoins des élèves, le programme d'informatique est subdivisé en deux périodes distinctes, chacune poursuivant des objectifs spécifiques. Cette structuration vise à offrir une progression logique et à permettre aux élèves de développer des compétences solides en informatique, tout en intégrant harmonieusement ces connaissances dans d'autres disciplines.

Au cours de la première période du programme d'informatique pour les élèves de la première année l'accent est mise sur l'acquisition des fondamentaux de l'informatique allant de l'architecture des ordinateurs au codage de l'information. Par la suite, une introduction à l'algorithmique et la programmation en Python, langage officiel du programme, ainsi que les notions et les bibliothèques informa-

tiques, permettent aux élèves d'acquérir les connaissances et les outils nécessaires pour une mise en œuvre pratique des concepts abordés.

La seconde période s'articule autour de la concrétisation des connaissances acquises en première période dans des contextes interdisciplinaires, renforçant les liens avec les autres disciplines connexes et la compréhension globale et la capacité à résoudre des problèmes concrets mathématiques et physiques tels que la résolution de differents types d'équations et le traitement d'images.

### 3.1.4 Exemple de Cours et TP

# Les fichiers

## 1 Introduction

Un fichier peut être vu comme un ensemble de bits stockés dans la mémoire pour une utilisation future et qui sont interprétés selon le type de fichier.

Il existe généralement deux types de fichiers :

- les fichiers textes qui contiennent une suite lisible de caractères.
- les fichiers binaires qui contiennent généralement une suite de bits, non lisible directement, destinés à être interprétés comme autre chose que des caractères de texte comme d'images, de sons, d'exécutables, …

On s'intéresse à la manipulation des fichiers texte.

## 2 Ouverture et fermeture de fichier

Tout fichier doit être ouvert avant de pouvoir accéder à son contenu en lecture et écriture.

La fonction $open(chemin\_fichier, mode)$ renvoie un objet fichier en utilisant deux arguments :

- Le premier est une chaîne de caractères représentant le chemin du fichier qui peut être :
  - ***relatif*** : basé sur le répertoire courant (généralement là où se trouve le programme en exécution).
  - ***absolu*** : emplacement dans la structure de système de fichiers, depuis la racine.
- Le deuxième argument est un caractère décrivant la façon dont le fichier est utilisé. Ce mode peut être :
  - ***'r'*** : (*read*) en lecture seulement (valeur par défaut);
  - ***'w'*** : (*write*) en écriture seulement en écrasant l'existant. Si le fichier n'existe pas, il sera créé;
  - ***'a'*** : (*append*) en écriture seulement à la fin du fichier. Si le fichier n'existe pas, il sera créé;
  - ***'x'*** : en création exclusive, pour prévenir l'écrasement ou l'ajout.

Pour fermer le fichier afin d'exploiter sa mise à jour et libérer les ressources système qu'il utilise on doit appeler la fonction $close()$.

### 2.1 Exemple

[1]:
```
# En utilisant un chemin relatif, le fichier se trouve dans le même répertoire␣
↪du programme

mon_fichier = open('todo.txt', 'r')
mon_fichier.close()
```

[2]:
```
# En utilisant un chemin absolu que l'on peut obtenir en utilisant la fonction␣
↪getcwd() du module os

import os
print(os.getcwd())
```

```
/Users/yussuf
```

[3]:
```
mon_fichier = open('/Users/yussuf/todo.txt', 'r')
mon_fichier.close()
```

## 3 Méthodes des objets fichiers

### 3.1 Lecture

Pour lire le contenu d'un fichier, on utilise la fonction *read*(*taille*) : elle renvoie *taille* octets de données sous la forme d'une chaîne de caractères. Quand le paramètre *taille* est omis, le contenu entier du fichier est lu et donné.

#### 3.1.1 Exemple

[4]:
```
# Ouverture

mon_fichier = open('todo.txt', 'r')

# Lecture du contenu entier
contenu = mon_fichier.read()

mon_fichier.close()

print(contenu)
```

```
Acheter du pain
Arroser les plantes
Payer la facture d'eau et d'électricité
```

[5]:
```
mon_fichier = open('todo.txt', 'r')

# 7 premier caractères seulement
```

```
contenu = mon_fichier.read(7)

mon_fichier.close()

print(contenu)
```

```
Acheter
```

Pour lire une seule ligne du fichier on utilise la fonction *readline*() qui, à chaque appel, renvoie la ligne suivante. Lorsque la fin du fichier est atteinte, la fonction renvoie une chaîne de caractères vide ''.

### 3.1.2 Exemple

[6]:
```
mon_fichier = open('todo.txt', 'r')

ligne1 = mon_fichier.readline()
ligne2 = mon_fichier.readline()
ligne3 = mon_fichier.readline()
ligne4 = mon_fichier.readline()


mon_fichier.close()

print("Première ligne : ", ligne1)
print("Deuxième : ", ligne2)
print("Troisième : ", ligne3)
print("Quatrième : ", ligne4)
```

```
Première ligne :  Acheter du pain

Deuxième :  Arroser les plantes

Troisième :  Payer la facture d'eau et d'électricité

Quatrième :
```

Pour construire une liste avec toutes les lignes d'un fichier, il suffit d'utiliser la fonction *readlines*(). Ainsi, on peut itérer sur les lignes à travers cette liste.

### 3.1.3 Exemple

[7]:
```
mon_fichier = open('todo.txt', 'r')

lignes = mon_fichier.readlines()

mon_fichier.close()

for ligne in lignes:
```

```
    print(ligne)
```

```
Acheter du pain

Arroser les plantes

Payer la facture d'eau et d'électricité
```

## 3.2 Écriture

La fonction *write*(*contenu*) écrit la chaîne de caractères *contenu* dans le fichier et renvoie le nombre de caractères écrits.

### 3.2.1 Exemple

[8]:
```
mon_fichier = open('todo.txt', 'a')

mon_fichier.write('Réserver le billet de train')

mon_fichier.close()
```

[9]:
```
mon_fichier = open('todo.txt', 'r')

contenu = mon_fichier.read()

mon_fichier.close()

print(contenu)
```

```
Acheter du pain
Arroser les plantes
Payer la facture d'eau et d'électricité
Réserver le billet de train
```

Pour écrire plusieurs lignes on utilise la fonction *writelines*(*liste_lignes*) qui prend la liste de lignes à écrire marquer de $\backslash n$ à la fin.

[10]:
```
mon_fichier = open('todo.txt', 'w')

mon_fichier.writelines(['Regarder le documentaire Planète Terre\n', 'Acheter du␣
 ↪lait\n', 'Imprimer le document X\n'])

mon_fichier.close()
```

[11]:
```
mon_fichier = open('todo.txt', 'r')

contenu = mon_fichier.read()
```

```
mon_fichier.close()

print(contenu)
```

```
Regarder le documentaire Planète Terre
Acheter du lait
Imprimer le document X
```

Dans le cas d'une grande liste de lignes sans $\backslash n$ et pour faciliter, on peut recourir à la méthode *join* en utilisant $\backslash n$ comme séparateur :

[12]:
```
liste_lignes = ['Regarder le documentaire Planète Terre', 'Acheter du lait',␣
↪'Imprimer le document X']

mon_fichier = open('todo.txt', 'w')

mon_fichier.write('\n'.join(liste_lignes))

mon_fichier.close()
```

[13]:
```
mon_fichier = open('todo.txt', 'r')

contenu = mon_fichier.read()

mon_fichier.close()

print(contenu)
```

```
Regarder le documentaire Planète Terre
Acheter du lait
Imprimer le document X
```

# 4 TP

## 4.1 Énoncé

***Exercie 1***

1. Créer une fonction $fusionner(f1, f2)$ qui permet de fusionner deux fichiers $f1$ et $f2$.
2. Créer une fonction $copier(fichier, chemin\_absolu)$ qui permet de copier un fichier dans un répertoire en utilisant un chemin absolu.

***Exercice 2***

1. Écrire une fonction $nombre\_lignes(fichier)$ qui renvoie le nombre de lignes dans un fichier.
2. Écrire une fonction $nombre\_mots(fichier)$ qui renvoie le nombre de mots dans un fichier.
3. Écrire une fonction $nombre\_caractères(fichier)$ qui renvoie le nombre de caractères dans un fichier.
4. Écrire une fonction $taille(fichier)$ qui renvoie la taille en *bit* d'un fichier.

***Exercice 3***

1. Créer un fichier *log.txt* pour sauvegarder les tâches effectuées en incluant la date et le temps d'ajout.
2. Écrire une fonction $task\_done(description)$ qui prend la description d'une tâche effectuée et l'ajoute à la fin du fichier *log*. La ligne ajoutée doit être de la forme $date\ \ temps\ \ description$. Pour cela on peut utiliser la fonction *now* du module *datetime* qui renvoie la date et le temps actuels.

*Exemple*

[14]:
```
from datetime import datetime

print(f'{datetime.now()} description\n')
```

```
2025-05-08 11:36:30.392252 description
```

4. Écrire une fonction $task(date)$ qui renvoie les tâches effectuées en une date donnée sous la forme $année - mois - jour$.

## 4.2 Résolution

[32]:
```
# Exercice 1

def fusionner(chemin1, chemin2):
    f1 = open(chemin1)
    f2 = open(chemin2)
    contenu = f1.read()+'\n'+f2.read()
    f1.close()
    f2.close()
    f3 = open('fusion.txt', 'x')
```

```
    f3.write(contenu)
    f3.close()

#fusionner('/Users/yussuf/fichier1.txt', '/Users/yussuf/fichier2.txt')

def copier(chemin_fichier, chemin_absolu):
    fichier = open(chemin_fichier)
    contenu = fichier.read()
    fichier.close()
    copie = open(chemin_absolu+'/fichier_copie.txt', 'x')
    copie.write(contenu)
    copie.close()

#copier('/Users/fichier.txt', '/Users/yussuf')

# Exercice 2

def nombre_lignes(fichier):
    x = open(fichier)
    nl = len(x.readlines())
    x.close()
    return nl

#print(nombre_lignes('todo.txt'))

def nombre_mots(fichier):
    x = open(fichier)
    lignes = x.read()
    x.close()
    return len(lignes.split())

#print(nombre_mots('todo.txt'))

def nombre_caractères(fichier):
    x = open(fichier)
    nc = x.read()
    x.close()
    return len(nc)

#print(nombre_caractères('todo.txt'))

def taille(fichier):
    return 8*nombre_caractères(fichier)

# Exercice 3

file = open('log.txt', 'x')
```

```
file.close()

def task_done(description):
    file = open('log.txt', 'a')
    file.write(f'{datetime.now()} {description}\n')
    file.close()

task_done('Cours envoyé')

def task(date):
    file = open('log.txt')
    lignes = file.readlines()
    for ligne in lignes:
        if ligne[0:10] == date:
            print(ligne)
    file.close()

task('2025-05-08')
```

```
2025-05-08 12:08:04.926158 Cours envoyé
```

## 3.2 1ère ÉCS

### 3.2.1 Structure

Deux classes, 1 et 2, composées respectivement de 33 élèves et 29 élèves, et réparties en deux groupes. L'enseignement de chaque groupe de classe se fait avec une fréquence de deux heures par quinzaine.

### 3.2.2 Objectif général

En première année, le programme d'informatique vise à la fois à développer des compétences spécifiques en informatique et à vérifier et interpréter numériquement des notions acquises dans d'autres disciplines mathématiques et économiques. Grâce à ce programme, Les élèves peuvent mettre en pratique leurs connaissances pour exploiter la puissance de l'informatique dans l'analyse économique, la modélisation et la prise de décision.

### 3.2.3 Contenu du programme

La première période du programme offre un aperçu général des systèmes informatiques, y compris l'architecture des ordinateurs et le codage de l'information, ensuite une introduction à l'algorithmique et la programmation en Python ainsi que les notions et les bibliothèques informatiques nécessaires aux calculs mathématiques sur les fonctions et les suites et leurs représentations graphiques.

La deuxième période est divisée en deux parties : La première est consacrée aux algorithmes avancées et au calcul scientifique; les algorithmes gloutons pour la résolution de problèmes notamment d'optimisation ainsi que des algorithmes élémentaires relatifs aux graphes finis. Dans la seconde, les élèves sont initiés à

l'analyse quantitative et à la simulation d'expériences aléatoires qui constituent deux composantes importantes pour la compréhension des phénomènes économiques. Cela développe chez les élèves les compétences nécessaires pour modéliser et analyser des scénarios économiques incertains.

### 3.2.4 Exemple de Cours et TP

# Les suites

## 1 Calcul du $n^{ème}$ terme

### 1.1 Suite explicite

Chaque terme de la suite est exprimé en fonction de $n$ et indépendamment des termes précédents :

$$U_n = f(n)$$

Pour calculer le $n^{ème}$ terme de la suite, il suffit de définir une fonction qui prend l'indice $n$ comme paramètre et calcule le terme correspondant.

*Exemple*

Soit $(U_n)_{n\in\mathbb{N}}$ une suite définie par :

$$U_n = \frac{1}{1+n}$$

[1]:
```
def U(n):
    return 1/(1+n)
```

[2]:
```
U(0)
```

[2]: 1.0

[3]:
```
U(9)
```

[3]: 0.1

On peut calculer et stocker les termes d'indices 0 à $n$ en utilisant une liste :

*Exemple*

[4]:
```
n = 3

termes = []                # initialisation de la liste des termes
for i in range(n+1):       # i va de 0 à n, donc n+1 termes vont être calculés
    termes.append(U(i))    # calcul et ajout du ième terme à la liste

print(termes)
```

```
[1.0, 0.5, 0.3333333333333333, 0.25]
```

[5]:
```
# ou bien en utilisant les listes en compréhension

n = 3

termes = [U(i) for i in range(n+1)]

termes
```

[5]:
```
[1.0, 0.5, 0.3333333333333333, 0.25]
```

## 1.2 Suite récurrente

### 1.2.1 Simple

Dans le cas d'une suite récurrente simple chaque terme de la suite s'obtient à partir du terme précédent :

$$U_{n+1} = f(U_n)$$

Pour calculer le $n^{ème}$ terme on initialise en affectant à une variable la valeur initiale de la suite, et, en utilisant une boucle qui se répète $n$ fois, on calcule le terme actuel en fonction du terme précédent, et la variable prend le terme qu'on vient de calculer.

*Exemple*

Soit la suite $(U_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} U_0 = 1 \\ U_{n+1} = 3U_n + 1 \end{cases}$$

[6]:
```
def U(n):
    x = 0                   # initialisation
    for _ in range(n+1):    # répétition n fois
        x = 3*x + 1         # mise à jour du terme
    return x
```

[7]:
```
U(1)
```

[7]:
```
4
```

[8]:
```
U(7)
```

[8]:
```
3280
```

En cas de besoin, on peut garder les termes précédents en utilisant une liste qui permettra de stocker progressivement chaque terme calculé.

*Exemple*

[9]:
```
def U(n):
    x = 1
```

```
    termes = [1]                # initialisation de la liste des termes
    for _ in range(n):          # pour calculer n termes
        x = 3*x + 1             # calcul et mise à jour
        termes.append(x)        # ajout du terme à la liste
    return termes

U(3)
```

[9]: `[1, 4, 13, 40]`

### 1.2.2 Double

Dans ce cas chaque terme est obtenu en combinant les deux termes précédents :

$$U_{n+2} = f(U_n, U_{n+1})$$

En utilisant la même technique vue dans le cas simple, on aura besoin d'initialiser en affectant à deux variables les deux valeurs initiales de la suite. Ensuite, en utilisant une boucle qui se répète $n$ fois, on calcule le terme actuel en fonction des termes précédents, et on met à jour les deux variables.

*Exemple*

Soit la suite $(U_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} U_0 = 0, U_1 = 1 \\ U_{n+2} = 3U_{n+1} + 2U_n + 1 \end{cases}$$

[10]:
```
def U(n):
    x1 = 0
    x2 = 1
    for _ in range(2, n+1):   # on saute les deux termes initiaux
        x = 3*x2 + 2*x1 + 1   # calcul du terme suivant
        x1 = x2               # mise à jour
        x2 = x
    return x

U(2)
```

[10]: `4`

Pareillement, on peut garder les termes précédents en utilisant une liste :

[11]:
```
def U(n):
    x1 = 0
    x2 = 1
    termes = [x1, x2]         # initialisation de la liste des termes
    for _ in range(2, n+1):
        x = 3*x2 + 2*x1 + 1   # calcul du terme suivant
```

```
        termes.append(x)      # ajout du terme à la liste
        x1 = x2               # mise à jour
        x2 = x
    return termes

U(3)
```

[11]: [0, 1, 4, 15]

## 2 Représentation graphique

Après avoir créé la liste des abscisses (indices des termes) et la liste des ordonnées (les termes), on peut représenter graphiquement la suite en utilisant la fonction $plot(indices, termes, marker)$ du module *matplotlib.pyplot* en utilisant le paramètre optionnel *marker* pour marquer les points de différents signes comme '.', 'o', 'x', ...

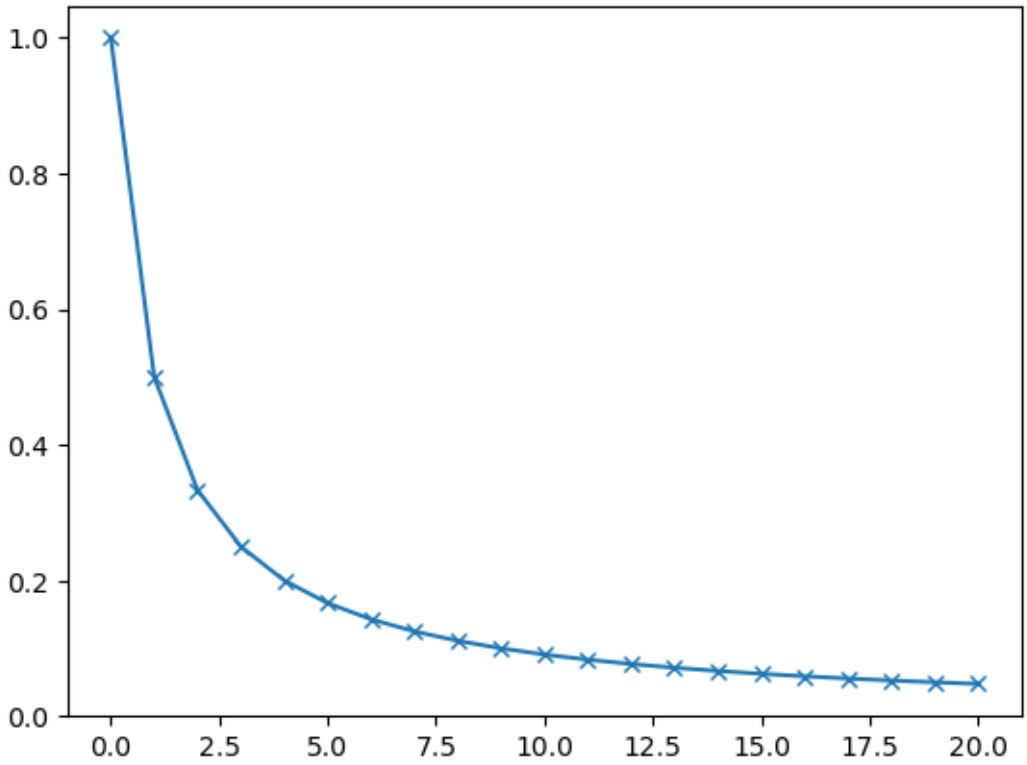
*Exemple*

On reprend la suite $(U_n)_{n\in\mathbb{N}}$ définie par :

$$U_n = \frac{1}{1+n}$$

[12]:
```
import numpy as np
import matplotlib.pyplot as mpl

def U(n):
    return 1/(1+n)

n = 20

# Création des indices
indices = np.arange(0, n+1, 1)

# Calcul des termes
termes = U(indices)

mpl.plot(indices, termes, marker='x')
```

[12]: [<matplotlib.lines.Line2D at 0x105de7750>]


1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

# 3 TP

## 3.1 Énoncé

***Exercice 1***

Soit $(U_n)_{n\in\mathbb{N}}$ une suite explicite définie par : $U_n = \frac{1}{2^n}$.

1. Créer une fonction qui calcule le $n^{ème}$ terme de la suite.
2. Créer une liste contenant les termes d'indices 0 à une borne choisie.
3. Tracer graphiquement la suite.
4. Calculer la limite de la suite et vérifier le résultat graphiquement.

***Exercice 2***

Soit $(U_n)_{n\in\mathbb{N}}$ une suite définie par : $U_n = (n-3)^2$.

1. Créer une fonction qui calcule le $n^{ème}$ terme de la suite.
2. Créer une liste contenant les termes d'indices 0 à une borne choisie.
3. Tracer graphiquement la suite.
4. Étudier la monotonie de la suite et vérifier le résultat graphiquement.

***Exercice 3***

Une usine assure, en 2025, une production de 100 000 articles. Elle s'engage à augmenter sa production de 3% chaque année.

1. Quelle est la nature de cette suite production?
2. Créer une fonction qui renvoie la production en une année donnée.
3. Créer une liste de nombres d'articles produits entre 2025 et 2035.
4. Représenter graphiquement la production entre 2025 et 2035.
5. Combien d'articles auront été fabriqués dans cette période?

***Exercice 4***

Soit la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_{n+2} = F_{n+1} + F_n \end{cases}$$

1. Créer une fonction pour calculer le $n^{ème}$ terme de la suite.
2. Stocker dans une liste les $\frac{F_n}{F_{n-1}}$ pour $n \in [2, 20]$.
3. Tracer graphiquement cette suites de rapports.
4. Tracer la ligne d'équation $f(x) = \varphi = \frac{1+\sqrt{5}}{2}$, pour $x \in [2, 20]$ (Le nombre d'or).
5. Que peut-on en déduire?

### 3.2 Résolution

[13]:
```
# Exercice 1

# 1

def U(n):
    return 1/2**n
```

[14]:
```
# 2

n = int(input('Une borne :'))

termes = [U(i) for i in range(n+1)]
```

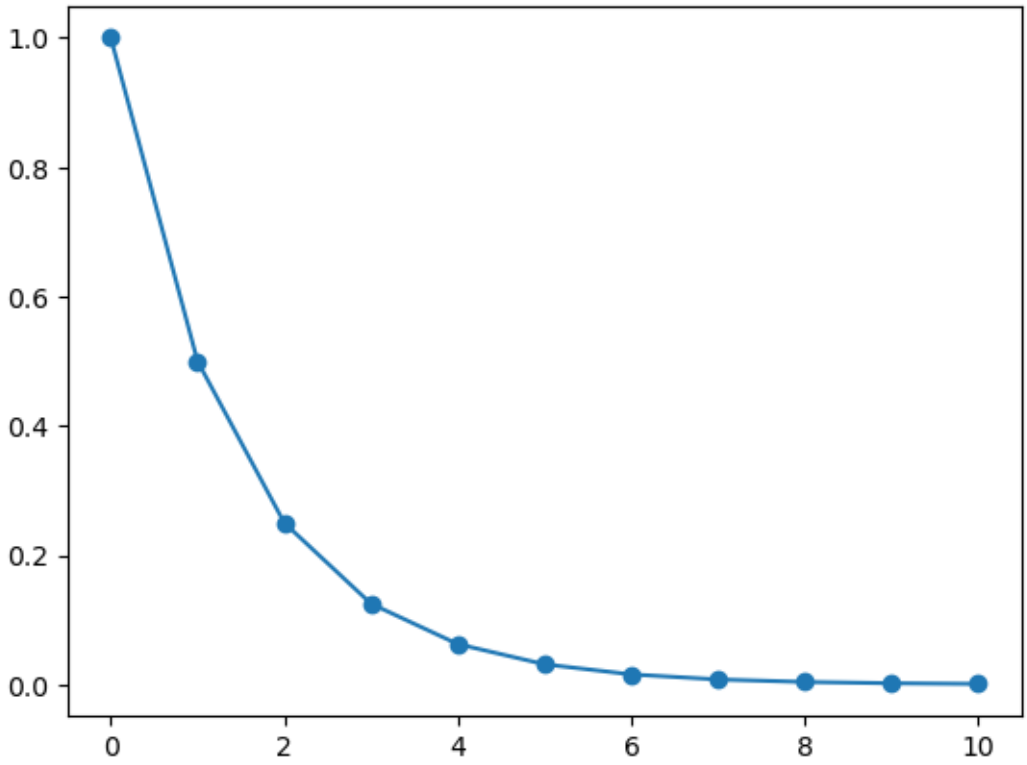
```
Une borne :10
```

[15]:
```
termes
```

[15]:
```
[1.0,
 0.5,
 0.25,
 0.125,
 0.0625,
 0.03125,
 0.015625,
 0.0078125,
 0.00390625,
 0.001953125,
 0.0009765625]
```

[16]:
```
# 3

indices = range(n+1)

mpl.plot(indices, termes, marker='o')
```

[16]:
```
[<matplotlib.lines.Line2D at 0x1068d29d0>]
```




4. On a $\lim\limits_{n\to\infty} U_n = 0$, ce qui est confirmé graphiquement.

[17]:
```
# Exercice 2

# 1

def U(n):
    return (n-3)**2
```

[18]:
```
# 2

n = int(input('Une borne :'))

termes = [U(i) for i in range(n+1)]
```

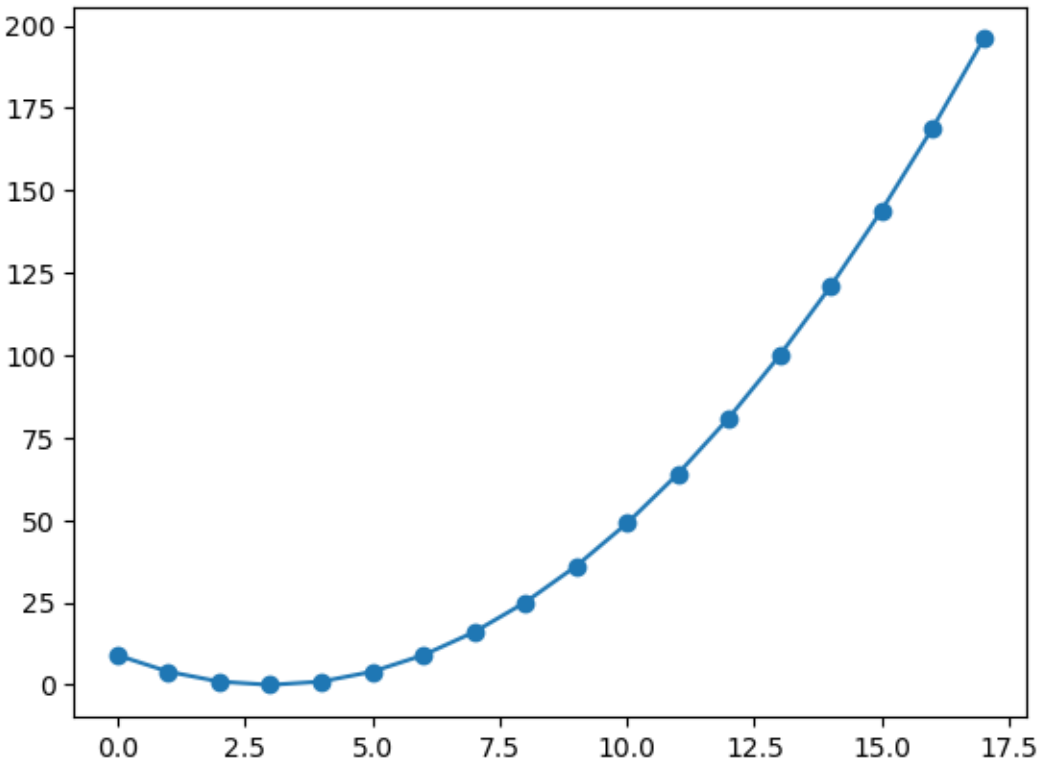
```
Une borne :17
```

[19]:
```
termes
```

[19]: `[9, 4, 1, 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196]`

[20]:
```
# 3

indices = range(n+1)

mpl.plot(indices, termes, marker='o')
```

[20]:
```
[<matplotlib.lines.Line2D at 0x106977890>]
```

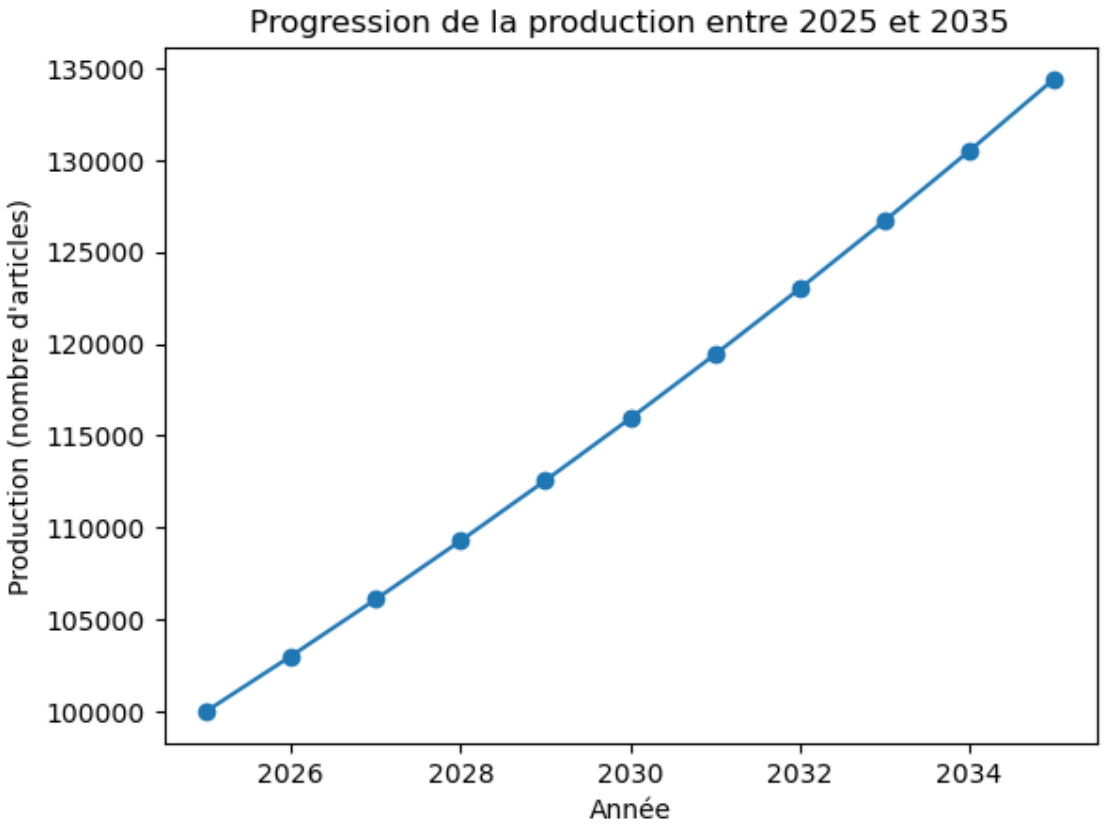



4. La suite est décroissante pour $n < 3$ et croissante sinon, ce qui est confirmé graphiquement.

[21]:
```
# Exercice 3

# 2

def production(an):
    n = an-2025    # nombre d'années à partir de 2025
    p0 = 100000
    return p0*1.03**n    # la production est une suite géométrique
```

[22]:
```
# 3

productions = [production(i) for i in range(2025, 2036)]
```

[23]:
```
# 4

années = range(2025, 2036)

mpl.plot(années, productions, marker='o')
mpl.xlabel('Année')
mpl.ylabel('Production (nombre d\'articles)')
mpl.title('Progression de la production entre 2025 et 2035')

mpl.show()
```




[24]:
```
print('Total des articles produits :', int(sum(productions)))
```

```
Total des articles produits : 1280779
```

[25]:
```
# Exercice 4

# 1

def F(n):
    assert isinstance(n, int)
```

```
    if n<=1:
        return n
    else:
        x0 = 0
        x1 = 1
        for _ in range(2, n+1):
            x = x0 + x1
            x0 = x1
            x1 = x
        return x

F(2)
```

[25]: 1

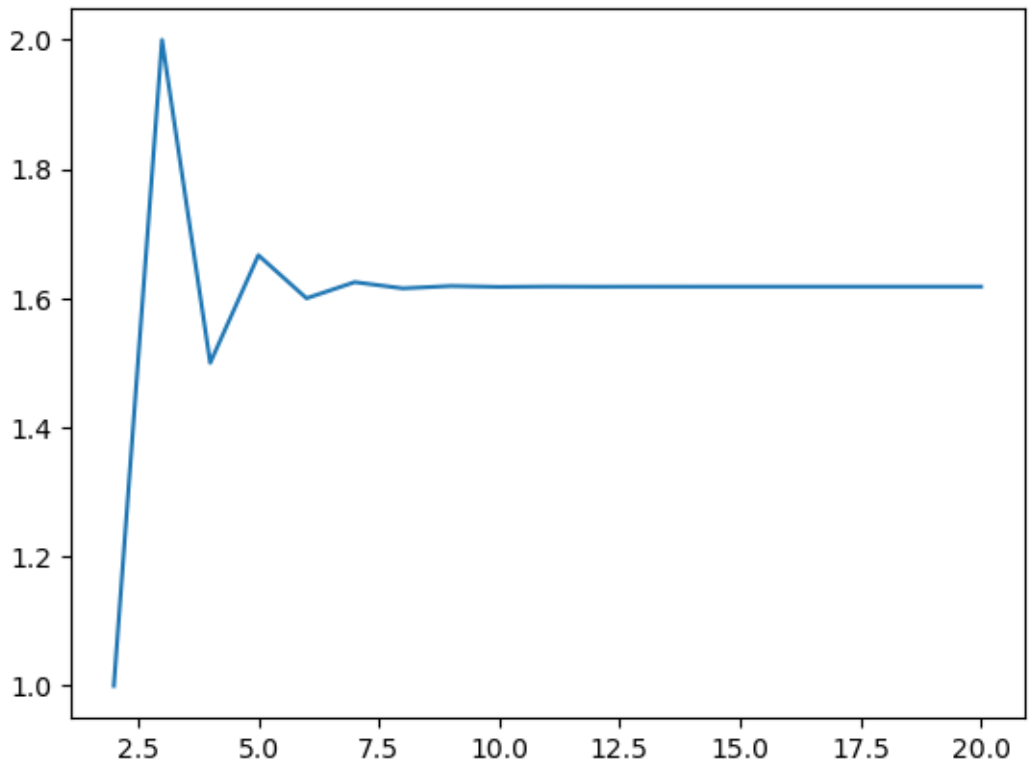
[26]:
```
F(10)
```

[26]: 55

[27]:
```
# 2

rapports = [F(n)/F(n-1) for n in range(2, 21)]
```

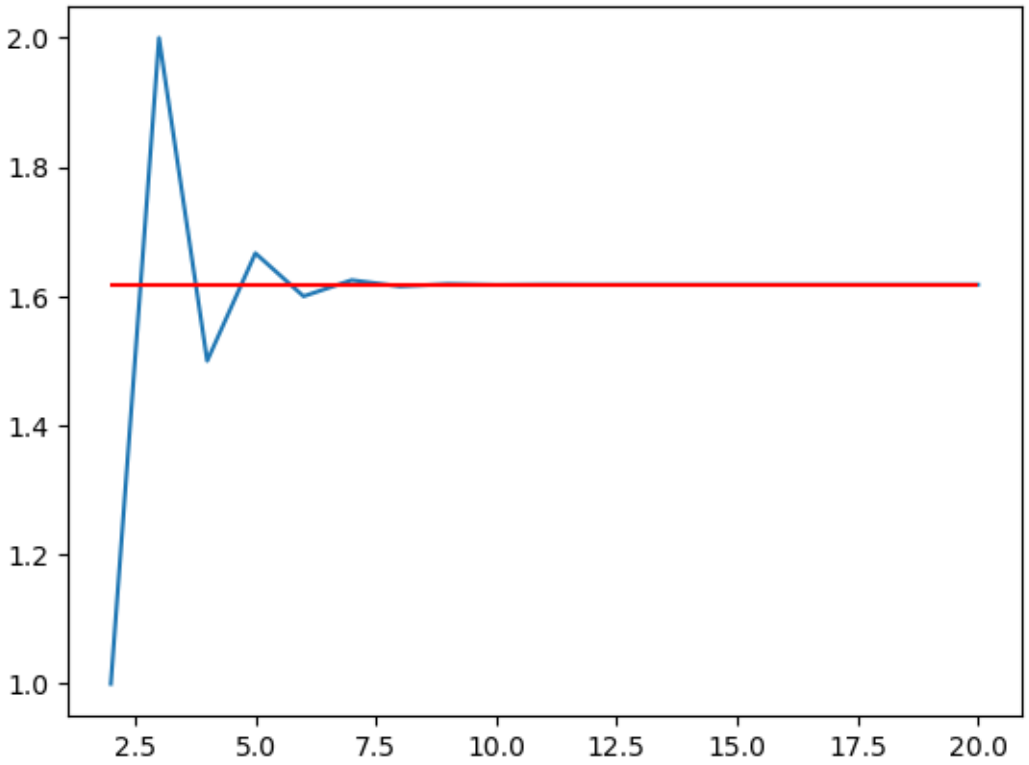
[28]:
```
# 3

indices = range(2, 21)

mpl.plot(indices, rapports)
```

[28]: [<matplotlib.lines.Line2D at 0x1068d0710>]




[29]:
```
# 4

mpl.plot(indices, rapports)

# tracer une ligne horizontale en nombre d'or entre 2 et 20
mpl.hlines((1+np.sqrt(5))/2, 2, 20, color='red')
```

[29]: <matplotlib.collections.LineCollection at 0x106a7b4d0>




5. On peut observer graphiquement que $\lim_{n\to\infty} \frac{F_n}{F_{n-1}} = \varphi$

## 3.3 2ème ÉCS

### 3.3.1 Structure

Deux classes, 1 et 2, composées respectivement de 25 élèves et 24 élèves, et réparties en deux groupes. L'enseignement de chaque groupe de classe se fait avec une fréquence de deux heures par quinzaine.

### 3.3.2 Objectif général

Le programme d'informatique en deuxième année des classes préparatoires aux grandes écoles de commerce vient compléter l'évolution de l'enseignement de l'informatique en première année.

### 3.3.3 Contenu du programme

La première période de ce programme concerne la gestion de bases de données. Puis, les élèves sont amenées à résoudre numériquement des problèmes plus avancés y compris des intégrales et des équations différentielles en utilisant les méthodes numériques appropriées et les outils informatiques adéquats.

La deuxième période de ce programme est consacrée aux techniques de modélisation avancées et de calcul scientifique allant de la simulation de lois de probabilités discrètes et continues, offrant aux élèves des méthodes pour vérifier et comparer des résultats théoriques, à la représentation graphique des fonctions de plusieurs variables.

### 3.3.4 Exemple de Cours et TP

# La librairie pandas et la représentation graphique

[1]:
```
import pandas
import matplotlib.pyplot as mpl

#importation des données
data_countries=pandas.read_csv('data_countries.csv')

#Un extrait de la table
data_countries.sample(7)
```

[1]:

| | Country | Continent | Year | LifeExp | Population | GDPPerCap |
|---|---|---|---|---|---|---|
| 363 | Croatia | Europe | 2007.0 | 75.748 | 4.493312e+06 | 14619.222720 |
| 1493 | Tanzania | Africa | 1957.0 | 42.974 | 9.452826e+06 | 698.535607 |
| 1607 | Uruguay | Americas | 1987.0 | 71.918 | 3.045153e+06 | 7452.398969 |
| 670 | Iceland | Europe | 1982.0 | 76.990 | 2.339970e+05 | 23269.607500 |
| 1605 | Uruguay | Americas | 1977.0 | 69.481 | 2.873520e+06 | 6504.339663 |
| 1422 | Sudan | Africa | 1962.0 | 40.870 | 1.118323e+07 | 1959.593767 |
| 298 | China | Asia | 2002.0 | 72.028 | 1.280400e+09 | 3119.280896 |

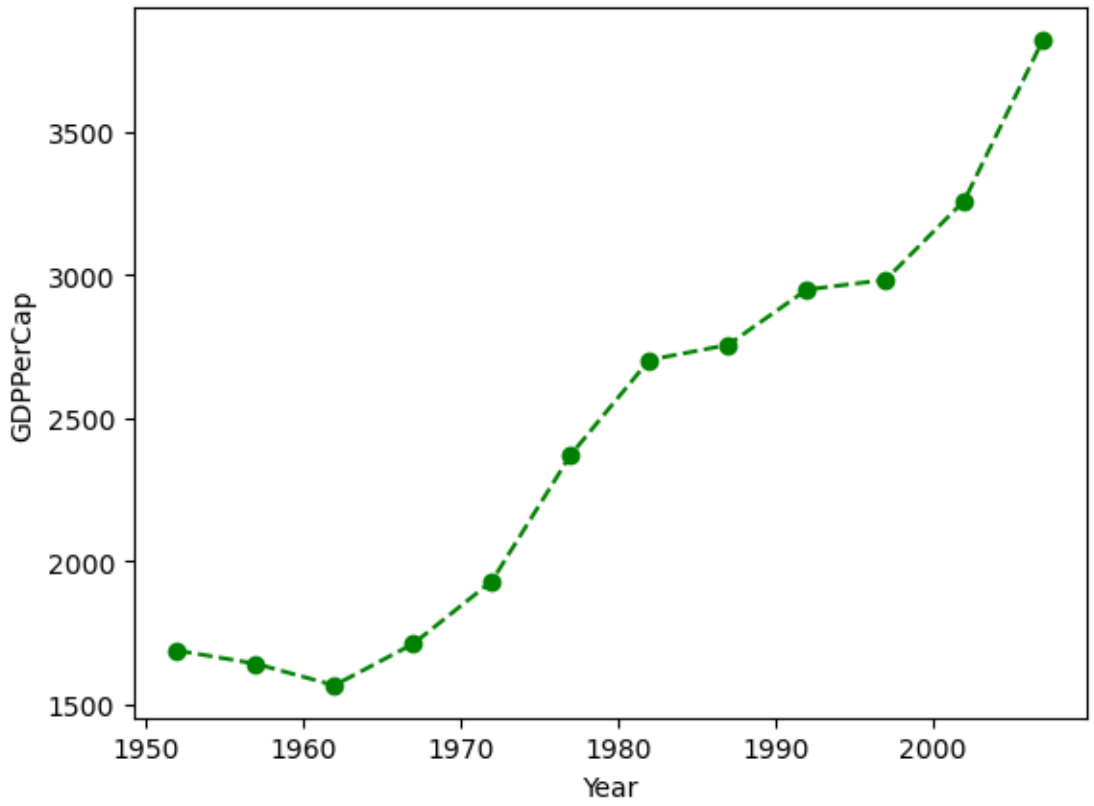
[27]:
```
#affichage des lignes concernant le maroc
country1='Morocco'
df1=data_countries[data_countries['Country']==country1]
df1
```

[27]:

| | Country | Continent | Year | LifeExp | Population | GDPPerCap |
|---|---|---|---|---|---|---|
| 988 | Morocco | Africa | 1952.0 | 42.873 | 9939217.0 | 1688.203570 |
| 989 | Morocco | Africa | 1957.0 | 45.423 | 11406350.0 | 1642.002314 |
| 990 | Morocco | Africa | 1962.0 | 47.924 | 13056604.0 | 1566.353493 |
| 991 | Morocco | Africa | 1967.0 | 50.335 | 14770296.0 | 1711.044770 |
| 992 | Morocco | Africa | 1972.0 | 52.862 | 16660670.0 | 1930.194975 |
| 993 | Morocco | Africa | 1977.0 | 55.730 | 18396941.0 | 2370.619976 |
| 994 | Morocco | Africa | 1982.0 | 59.650 | 20198730.0 | 2702.620356 |
| 995 | Morocco | Africa | 1987.0 | 62.677 | 22987397.0 | 2755.046991 |
| 996 | Morocco | Africa | 1992.0 | 65.393 | 25798239.0 | 2948.047252 |
| 997 | Morocco | Africa | 1997.0 | 67.660 | 28529501.0 | 2982.101858 |
| 998 | Morocco | Africa | 2002.0 | 69.615 | 31167783.0 | 3258.495584 |
| 999 | Morocco | Africa | 2007.0 | 71.164 | 33757175.0 | 3820.175230 |

[3]:
```
#Exprimer graphiquement le PIB en fonction des années

mpl.plot(df1['Year'], df1['GDPPerCap'], 'go--')
mpl.xlabel('Year')
mpl.ylabel('GDPPerCap')
```

[3]:
```
Text(0, 0.5, 'GDPPerCap')
```



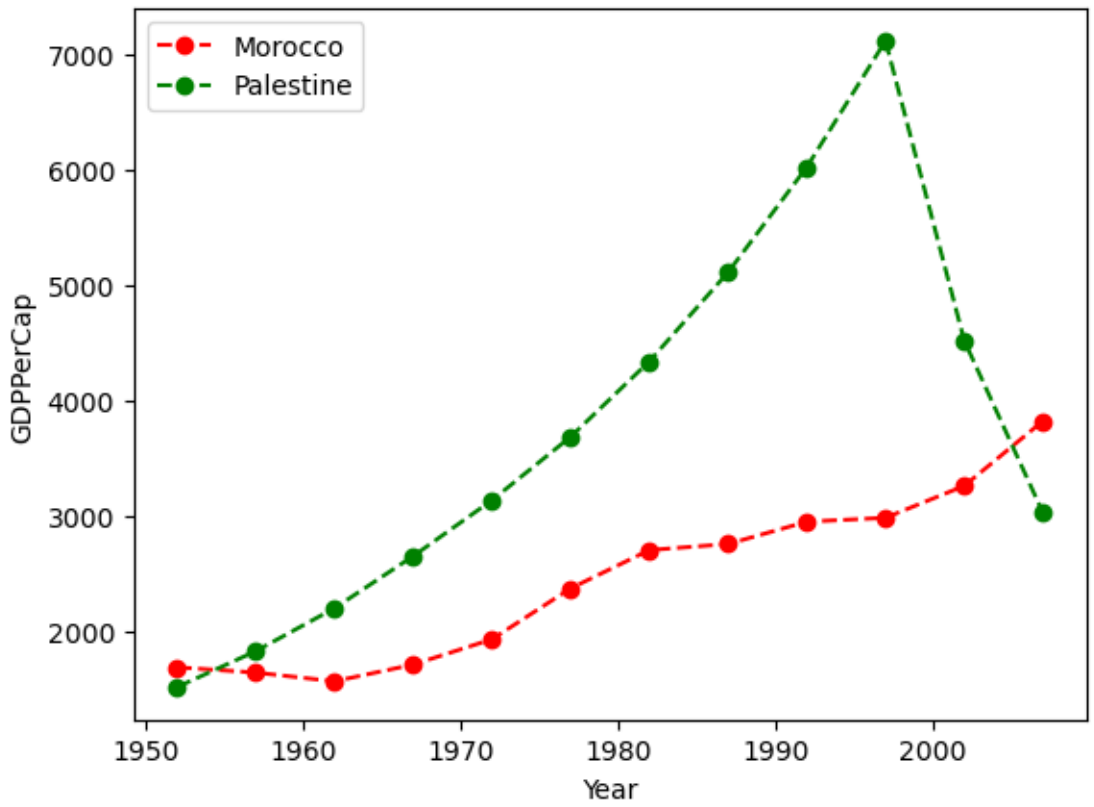

[4]:
```
#Affichage des payés concernés
data_countries['Country'].unique()
```

[4]:
```
array(['Afghanistan', 'Albania', 'Algeria', 'Angola', 'Argentina',
       'Australia', 'Austria', 'Bahrain', 'Bangladesh', 'Belgium',
       'Benin', 'Bolivia', 'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'Bulgaria', 'Burkina Faso', 'Burundi', 'Cambodia', 'Cameroon',
       'Canada', 'Central African Republic', 'Chad', 'Chile', 'China',
       'Colombia', 'Comoros', 'Congo', 'Costa Rica', 'Croatia', 'Cuba',
       'Czech Republic', 'Denmark', 'Djibouti', 'Dominican Republic',
       'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
       'Ethiopia', 'Finland', 'France', 'Gabon', 'Gambia', 'Germany',
       'Ghana', 'Greece', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Haiti',
```

```
'Honduras', 'Hong Kong, China', 'Hungary', 'Iceland', 'India',
'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Italy', 'Ivory Coast',
'Jamaica', 'Japan', 'Jordan', 'Kenya', 'Korea, Dem. Rep.',
'Korea, Rep.', 'Kuwait', 'Lebanon', 'Liberia', 'Libya',
'Madagascar', 'Malawi', 'Malaysia', 'Mali', 'Mauritania',
'Mauritius', 'Mexico', 'Mongolia', 'Montenegro', 'Morocco',
'Mozambique', 'Myanmar', 'Namibia', 'Nepal', 'Netherlands',
'New Zealand', 'Nicaragua', 'Niger', 'Nigeria', 'Norway', 'Oman',
'Pakistan', 'Palestine', 'Panama', 'Paraguay', 'Peru',
'Philippines', 'Poland', 'Portugal', 'Puerto Rico', 'Reunion',
'Romania', 'Rwanda', 'Sao Tome and Principe', 'Saudi Arabia',
'Senegal', 'Serbia', 'Sierra Leone', 'Singapore',
'Slovak Republic', 'Slovenia', 'Somalia', 'South Africa', 'Spain',
'Sri Lanka', 'Sudan', 'Swaziland', 'Sweden', 'Switzerland',
'Syria', 'Taiwan', 'Tanzania', 'Thailand', 'Togo',
'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Uganda',
'United Kingdom', 'United States', 'Uruguay', 'Venezuela',
'Vietnam', 'Yemen, Rep.', 'Zambia', 'Zimbabwe'], dtype=object)
```

[5]:
```
country2='Palestine'
df2=data_countries[data_countries['Country']==country2]

#Tracer plusieurs courbes en même temps
mpl.plot(df1['Year'], df1['GDPPerCap'], 'ro--', label=country1)
mpl.plot(df2['Year'], df2['GDPPerCap'], 'go--', label=country2)
mpl.xlabel('Year')
mpl.ylabel('GDPPerCap')
mpl.legend() #pour distinguer les courbes
```

[5]:
```
<matplotlib.legend.Legend at 0x12af0fcd0>
```




[6]:
```
#affichage des statistiques de l'année 2007
df_2007 = data_countries[data_countries['Year']==2007]
df_2007
```

[6]:

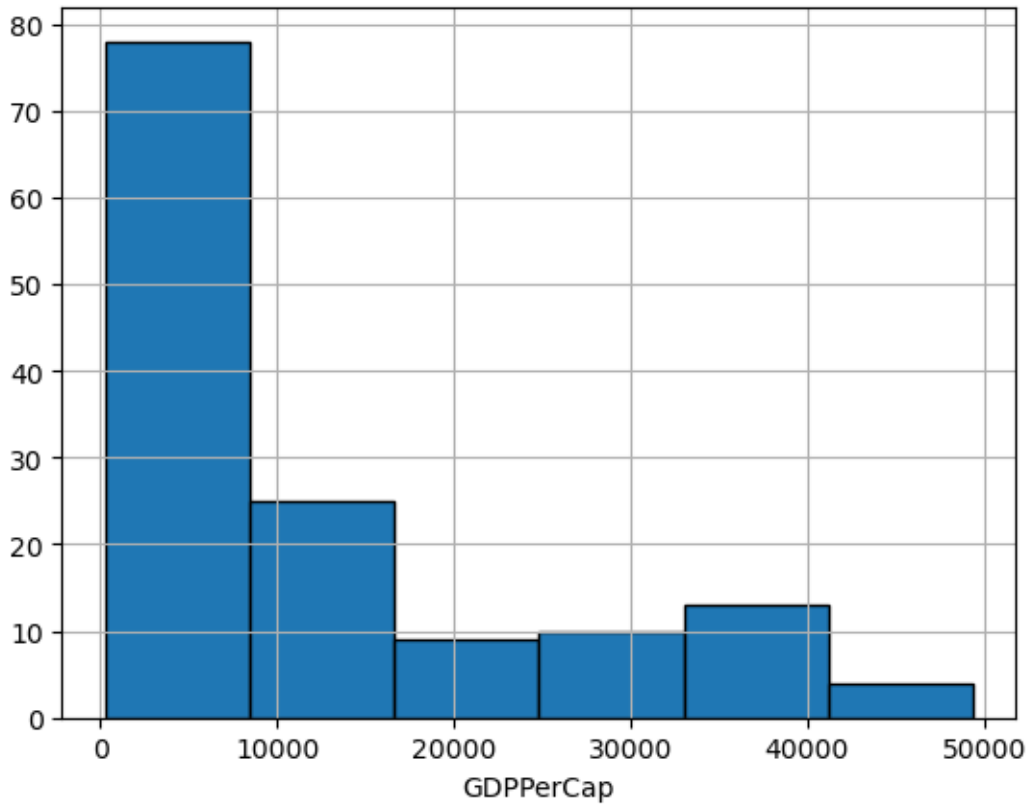
| | Country | Continent | Year | LifeExp | Population | GDPPerCap |
|---|---|---|---|---|---|---|
| 11 | Afghanistan | Asia | 2007.0 | 43.828 | 31889923.0 | 974.580338 |
| 23 | Albania | Europe | 2007.0 | 76.423 | 3600523.0 | 5937.029526 |
| 35 | Algeria | Africa | 2007.0 | 72.301 | 33333216.0 | 6223.367465 |
| 47 | Angola | Africa | 2007.0 | 42.731 | 12420476.0 | 4797.231267 |
| 59 | Argentina | Americas | 2007.0 | 75.320 | 40301927.0 | 12779.379640 |
| … | … | … | … | … | … | … |
| 1623 | Venezuela | Americas | 2007.0 | 73.747 | 26084662.0 | 11415.805690 |
| 1635 | Vietnam | Asia | 2007.0 | 74.249 | 85262356.0 | 2441.576404 |
| 1647 | Yemen, Rep. | Asia | 2007.0 | 62.698 | 22211743.0 | 2280.769906 |
| 1659 | Zambia | Africa | 2007.0 | 42.384 | 11746035.0 | 1271.211593 |
| 1671 | Zimbabwe | Africa | 2007.0 | 43.487 | 12311143.0 | 469.709298 |

[139 rows x 6 columns]

[7]:
```
#Expression des statistiques en ce qui concerne le PIB en 2007 sous forme␣
↪d'histogramme
```

```
mpl.hist(df_2007['GDPPerCap'], edgecolor='black', bins=6) #bins pour spécifier␣
 ↪le nombre des barres
mpl.grid()
mpl.xlabel('GDPPerCap')

#Calcul de la moyenne et de l'écart-type

print("La moyenne en 2007 était ", df_2007['GDPPerCap'].mean(), "et␣
 ↪l''écart-type ", df_2007['GDPPerCap'].std())

#Calcul de la médiane

print("La médiane  en 2007 était ", df_2007['GDPPerCap'].median())
```

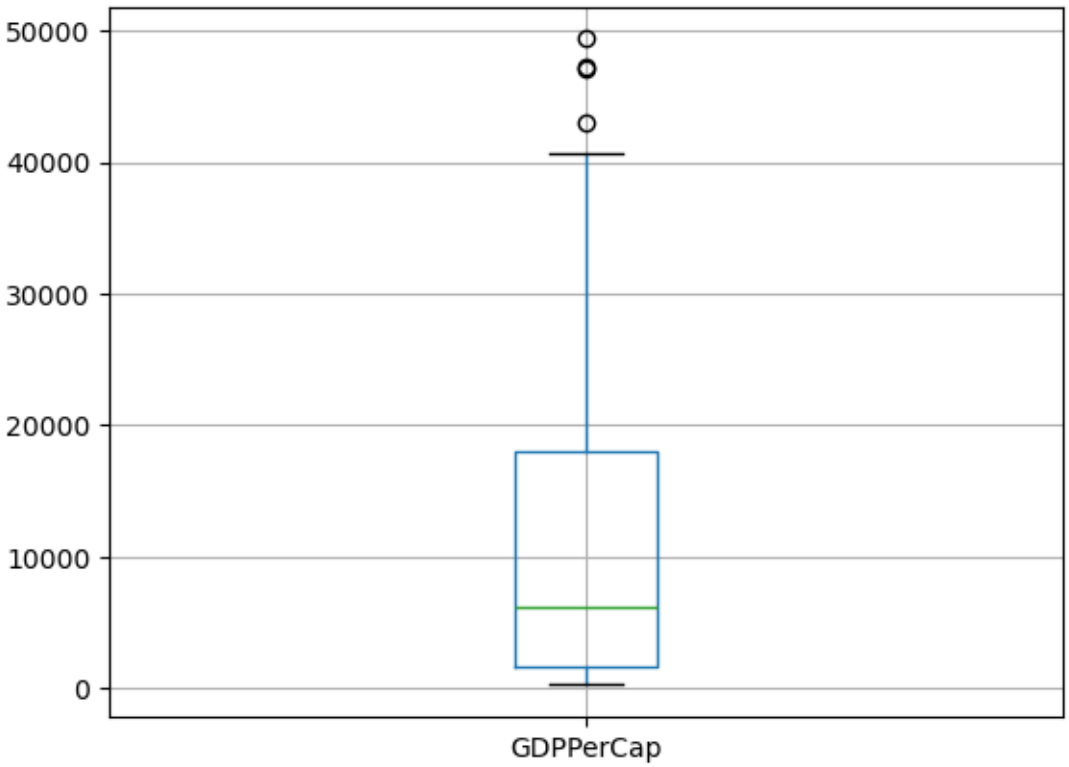
```
La moyenne en 2007 était  11711.115338724461 et l''écart-type
12898.583933569626
La médiane  en 2007 était  6223.367465
```



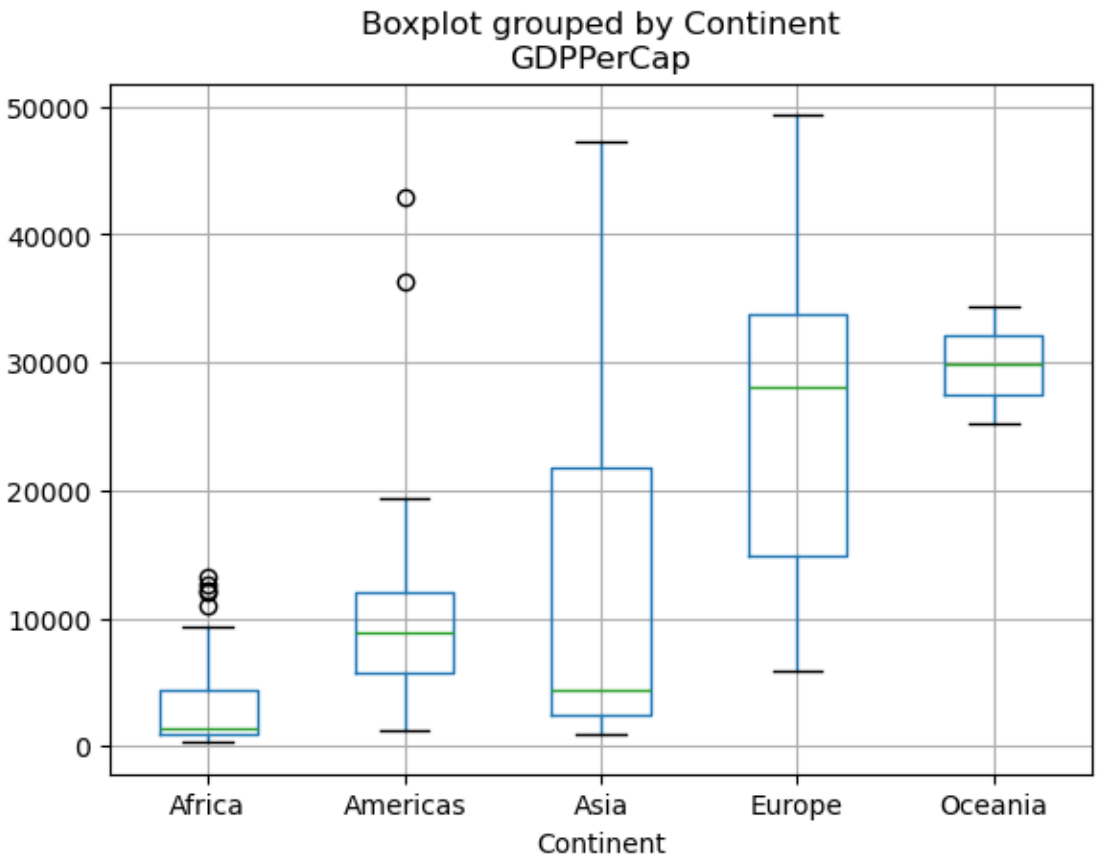

[8]:
```
#représentation en utilisant les quartiles avec la boîte à moustaches
```

```
df_2007.boxplot(column=['GDPPerCap'], grid=True)
```

[8]: <Axes: >




[9]:
```
#représentation par continent
df_2007.boxplot(by='Continent', column=['GDPPerCap'], grid=True);
```


Boxplot grouped by Continent
GDPPerCap
50000
40000
30000
20000
10000
0
Africa
Americas
Asia
Europe
Oceania
Continent

# 4
# Difficultés rencontrées

L'enseignement d'un groupe de classe avec une fréquence de seulement deux heures par semaine ou, pire encore, deux semaines présente un certain nombre de défis, tant sur le plan pédagogique qu'organisationnel. Cette contrainte de temps impacte directement la dynamique d'apprentissage, la gestion de la progression, et l'engagement des élèves.

La principale difficulté réside dans le temps restreint qui ne permet pas toujours de développer les compétences de manière approfondie. Il devient difficile d'installer des routines pédagogiques efficaces, de différencier les apprentissages ou de proposer des projets ambitieux. Cette limitation de temps oblige à faire des choix parfois frustrants dans les contenus abordés et les méthodes utilisées.

Les élèves peuvent rapidement oublier les notions abordées précédemment, surtout s'ils ne sont pas sollicités entre-temps. Cela oblige souvent l'enseignant à consacrer un temps important à la réactivation des connaissances, ce qui ralentit considérablement la progression globale.

En outre, les élèves peuvent avoir du mal à rester engagés ou à comprendre l'importance de ces heures, surtout si la discipline enseignée est perçue comme secondaire.

# 5
# Conclusion

Ce stage a été une expérience enrichissante tant sur le plan professionnel que personnel.

Il m'a permis non seulement de mieux comprendre la rigueur attendue à ce niveau d'études, mais aussi de voir comment l'informatique peut être enseignée de manière vivante.

Il m'a donné l'opportunité d'enrichir mes compétences pédagogiques et disciplinaires.

Cette immersion a renforcé mon intérêt de poursuivre dans cette voie, avec le désir de contribuer, moi aussi, à former les leaders de demain.