Cédric Cognard
Paul Van-Elsue

Python

Compte rendu des séances de TPs

Sommaire

I. TP 1 : Première approche

Hello World, menu console et manipulation de fichier

Pour commencer nous allons simplement réaliser un Hello World et un menu permettant de de manipuler un fichier contenant des informations sur des étudiants.

Pour tout ce qui est affichage dans la console un simple `print("txt")` suffit et pour récupérer les choix de l'utilisateur nous utilisons `input("txt")`. En fonction du choix nous allons appeler la fonction correspondante.

```
def menu(self):
    choice = 0
    while(choice != 9):
        while((choice < 1 or choice > 4) and choice != 9):
            print("'------------ Menu ------------'")
            print("1. Choisir un nom de fichier")
            print("2. Ajouter un texte")
            print("3. Afficher le fichier complet")
            print("4. Vider le fichier")
            print("9. Quitter le programme")
            choice = int(input("Choix : "))
        if(choice == 1):
            self.choose_filename()
        elif(choice == 2):
            self.add_text()
        elif(choice == 3):
            self.print_file()
        elif(choice == 4):
            self.empty_file()
        if(choice != 9):
            choice = 0
```

Ce menu va permettre à l'utilisateur de spécifier sur quel fichier il va travailler, pour cela on va venir prendre en input un chemin vers le fichier que l'on va placer dans l'attribut *filename* de notre classe.

Pour modifier le fichier nous utilisons les exemples fournis dans le sujet, comme par exemple avec la fonction `add_text()` qui permet d'écrire à la fin du fichier

```
def add_text(self):
    if(self.filename == ""):
        print("Merci d'entrer un nom de fichier.")
    else:
        text_to_add = input("Entrez le texte à ajouter : ")
        file = open(self.filename, 'a')
        file.write(text_to_add)
        file.close()
```

L'option *'a'* dans *open* nous place en bout de fichier, il n'y plus qu'à y mettre l'input utilisateur.

Classes Date et Étudiant

On souhaite désormais créer deux classe que l'on pourra utiliser pour ajouter des entrées dans notre document. Une première nous permettra d'utiliser les dates et une seconde qui regroupera toutes les informations sur les étudiants.
Pour cela, nous devons créer une fonction `__init__` qui initialisera nos attributs.

```
class Date:
   jour = 0
   mois = 0
   annee = 0

   def __init__(self, jour, mois, annee):
       self.jour = jour
       self.mois = mois
       self.annee = annee
```

Dans cette classe nous allons surcharger les fonctions `__eq__` (equals) , `__lt__` (lower than) et `__str__` (renvoie sous forme d'un string) de la façon suivante:

```
def __eq__(self, o: object):
   if(self.jour == o.jour and self.mois == o.mois and self.annee ==
o.annee):
       return True
   return False

def __lt__(self, other) -> bool:
   if(self.annee < other.annee):
       return True
   elif(self.annee == other.annee):
       if(self.mois < other.mois):
           return True
       elif(self.mois == other.mois):
           if(self.jour < other.jour):
               return True
   return False

def __str__(self):
   return self.jour + '/' + self.mois + '/' + self.annee
```

En faisant ces surcharges nous pouvons désormais comparer deux dates avec un égal ou un < (inférieur à) et nous avons donner un format de date avec la méthode `__str__`.

Nous allons alors créer la classe `Etudiant` de la même façon en ajoutant des méthodes qui permettent de créer l'adresse mail et de calculer l'âge en fonction de la date de naissance.

Exemple :

```
def adresslec(self):
    adress = self.nom + '.' + self.prenom + "@etu.univ-tours.fr"
```

Et enfin nous avons quelques getters et quelques méthodes utiles. Nous avons aussi une classe `EtudiantReader` pour lire le fichier étudiant fourni dans le TP.

Au final toutes les consignes demandées sont parfaitement fonctionnelles et nous pouvons passer au TP suivant.

II. TP2 : Tkinter

1. Introduction

Ce TP sert de première approche à la mise en place d'interface graphique en Python en utilisant la bibliothèque **Tkinter**.
Tkinter est la bibliothèque d'interface graphique de base en Python. Pour commencer à l'appréhender avec ce TP nous allons réaliser une calculatrice.

2. Mise en place de l'IHM

Pour commencer il faut importer la bibliothèque **Tkinter** et instancier notre fenêtre qui va l'utiliser de la façon suivante:

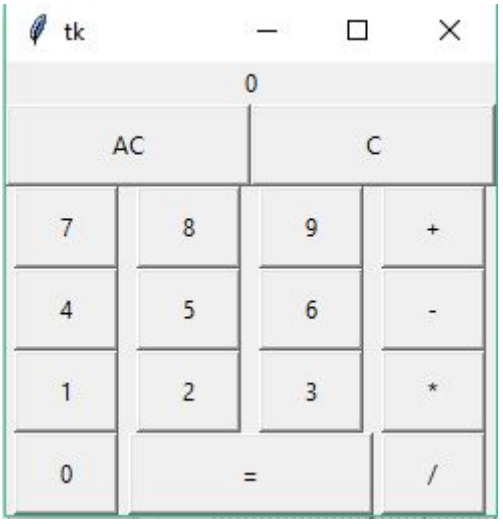
```
from tkinter import *

class Calculatrice:
   window = Tk()
```

Ainsi à partir de maintenant nous pouvons utiliser la bibliothèque graphique pour notre objet `window`, dans notre fonction `__init__` (constructeur) nous allons créer tous les boutons de notre calculatrice en utilisant la méthode `Button` de **Tkinter** de la façon suivante:

```
b = Button(self.window, text=0, command=lambda: self.number_function(0),
width=6, height=2)
b.grid(row=5, column=0)
```

Dans cet exemple nous construisons le bouton 0 de la calculette, en le plaçant sur la ligne 5 et la colonne 0 de notre `grid` ainsi ce bouton se situera en bas à gauche de la fenêtre.
La fonction `number_function` permet de remplir notre *String* contenant l'ensemble du calcul et d'afficher la valeur du bouton appuyé sur la calculatrice.

On fait de même pour tous les boutons de la calculatrice, on obtient au final cette fenêtre :



3. Méthodes de calcul

Maintenant que nous avons une calculatrice on aimerait qu'elle fasse des calculs, nous allons donc créer des méthodes pour réaliser les différentes opérations (addition, multiplication, soustraction, division).

```
def plus_function(self):
    self.calcul += "+"
    self.label.config(text=self.calcul)
```

Par exemple ici il s'agit de notre fonction pour l'addition on ajoute le caractère "+" dans la *String* contenant le calcul complet et la seconde ligne permet d'actualiser l'affichage dans notre fenêtre.

Pour calculer le contenu de notre *String* de calcul nous utilisons la fonction `eval()`. Cependant il faut prendre en compte certains calculs impossibles comme la division par 0. Pour cela nous allons mettre une **Exception** dans notre fonction appelée lors de l'appui sur le bouton '='.

```
def equal_function(self):
    try:
        self.calcul = str(eval(self.calcul))
        self.label.config(text=self.calcul)
    except ZeroDivisionError:
        self.label.config(text="ERROR, division by zero impossible.")
        self.calcul = ""
    except SyntaxError:
        self.label.config(text="Syntax error.")
        self.calcul = ""
```

Ainsi, si l'utilisateur essaye de diviser par 0 ou s'il ya une erreur de syntax le message correspondant sera affiché. Autrement le résultat du calcul apparaîtra sur la calculette.

III. TP3 : Exceptions et chiffrement

Introduction

Dans un premier temps, nous devions essayer les exceptions en Python. Très similaire aux autres langages, nous avons laissé quelques exemples de temps en temps dans nos programmes mais nous l'avons assez peu expérimenté. Nous sommes assez rapidement passé sur le hashage.

Le hashage

Pour commencer nous avons créé un système d'utilisateur avec login et mot de passe. Une classe *UserCrypto* (dans le TP3_bis, le TP3 étant des expérimentations diverses) permet de créer un nouvel utilisateur et écrit ce dernier dans un fichier texte. Pour pas que quelqu'un puisse récupérer le mot de passe en clair dans le fichier nous hashons son mot de passe en SHA512, l'algorithme de hashage le plus solide de nos jours (sans considérer les versions avec des blocs plus gros bien sûr).

Nous avons rajouté le salage pour plus de sécurité :

```
salt_password = username + password + username + self.salt
self.password = hashlib.sha512(salt_password.encode()).hexdigest()
passfile.write(username + ":" + self.password + '\n')
```

Ensuite, nous avons utilisé ce système d'utilisateur pour lui créer un menu et demander de chiffrer puis déchiffrer un fichier. Nous avons fait l'expérimentation avec un fichier texte et un chiffrement AES réputé pour être l'algorithme de chiffrement symétrique le plus sûr du moment. Deux méthodes ont donc été faites, encryptFile et uncryptFile. Au final c'est très fonctionnel et très puissant.

```
def encryptFile(self, filepath):
    file = open(filepath, "rb")
    data = b""
    lines = file.readlines()
    for line in lines:
        data = data + line + b"\n"

    cipher = AES.new(base64.b64decode(self.password)[:16], AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(data)

    file_out = open(filepath + ".encrypted", "wb")
    [file_out.write(x) for x in (cipher.nonce, tag, ciphertext)]

    file.close()
    file_out.close()

def uncryptFile(self, filepath):
    file_in = open(filepath, "rb")
    file_out = open(filepath.split(".encrypted")[0], "wb")

    nonce, tag, ciphertext = [file_in.read(x) for x in (16, 16, -1)]
    cipher = AES.new(base64.b64decode(self.password)[:16], AES.MODE_EAX, nonce)

    data = cipher.decrypt_and_verify(ciphertext, tag)
    data = data.split(b"\n")
    [file_out.write(x) for x in data]

    file_in.close()
    file_out.close()
```

IV. TP4 : Matplotlib

1. Introduction

Matplotlib est la bibliothèque la plus utilisée pour tracer des graphes en Python, ses applications sont innombrables en particulier dans le domaine scientifique et de la recherche. Dans ce TP nous allons apprendre à utiliser les fonctionnalités de base de cette bibliothèque et tracer différents type de graphiques.

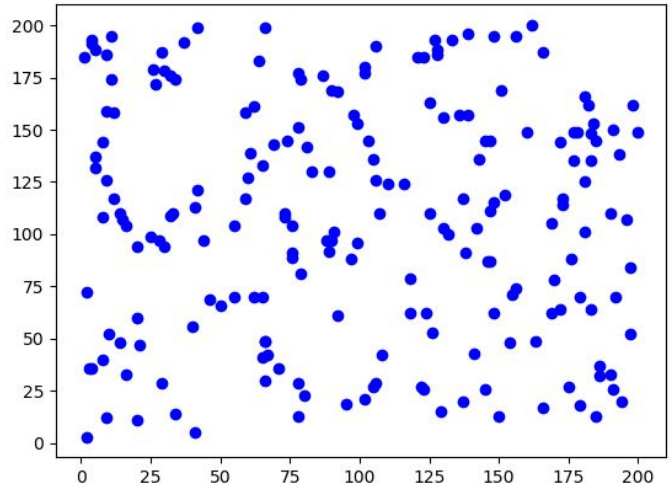
2. Nuage de points

Nous allons tracer un nuage de points générés aléatoirements, pour cela il nous faut des tableaux contenant les positions en X et en Y de chaque point.
On remplit ces tableaux avec des valeurs aléatoires (ici comprises entre 0 et 200) de la façon suivante :

```
random_numbers_X = []
random_numbers_Y = []

for i in range(200):
   random_numbers_X.append(random.randint(0, 200))
   random_numbers_Y.append(random.randint(0, 200))
```

Une fois qu'on a ces données il est possible de les voir graphiquement grâce à la fonction plot de **Matplotlib**. Cette fonction dispose de différents paramètres, on peut spécifier ce que l'on veut prendre en abscisse, en ordonnée, si nous voulons relier les points ou non ou encore la couleur de ces derniers, etc...

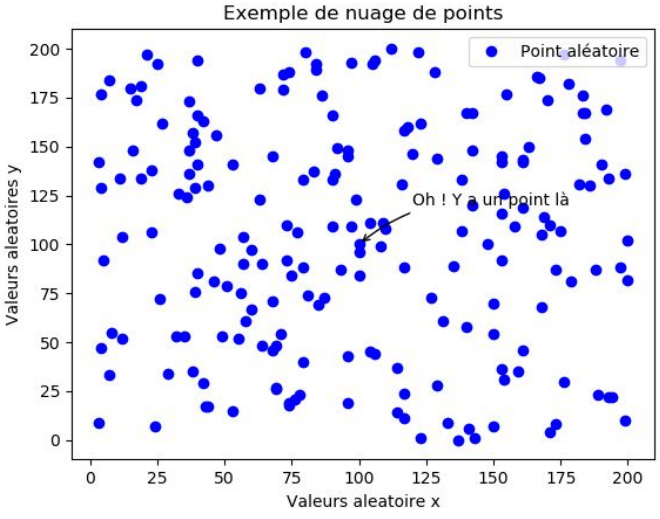
Pour afficher notre graphe nous devons appeler la fonction `show()`.



Voila notre nuage de points. Cependant on remarque que notre label “Point aléatoire” dans le label de notre fonction plot ne s’affiche pas. C’est parce que nous n’avons pas appelé la fonction `legend()` de **Matplotlib**. Il est possible de rajouter encore beaucoup d’annotations et de légendes autour de ce graphe.

Exemples :

```
plt.annotate('Oh ! Y a un point là', xy=(100, 100), xytext=(120, 120),
arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))
plt.xlabel('Valeurs aleatoire x')
plt.ylabel('Valeurs aleatoires y')
plt.title("Exemple de nuage de points")
```



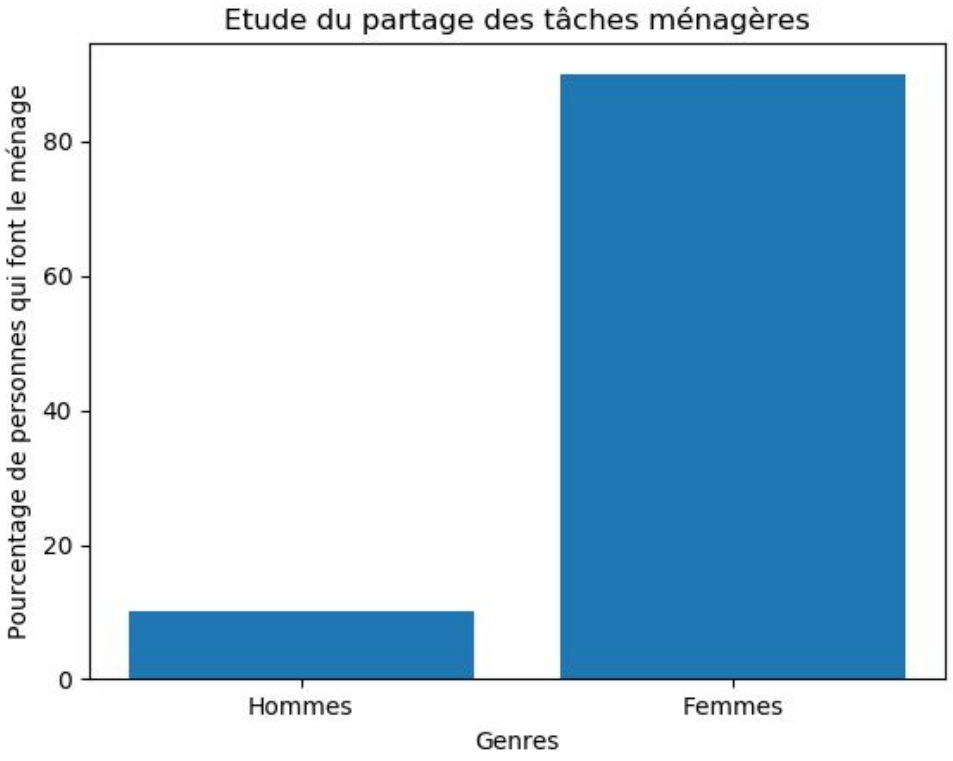
3. Histogramme

Pour tracer un histogramme on ne fait plus appel à la fonction plot mais à la fonction bar.

```
x = np.arange(2)
plt.bar(x, height=[10, 90])
```

De cette façon nous aurons 2 barres avec comme valeurs 10 pour la première et 90 pour la seconde. De la même façon que précédemment il est possible de rajouter des légendes sur le graphe.

On obtient au final:



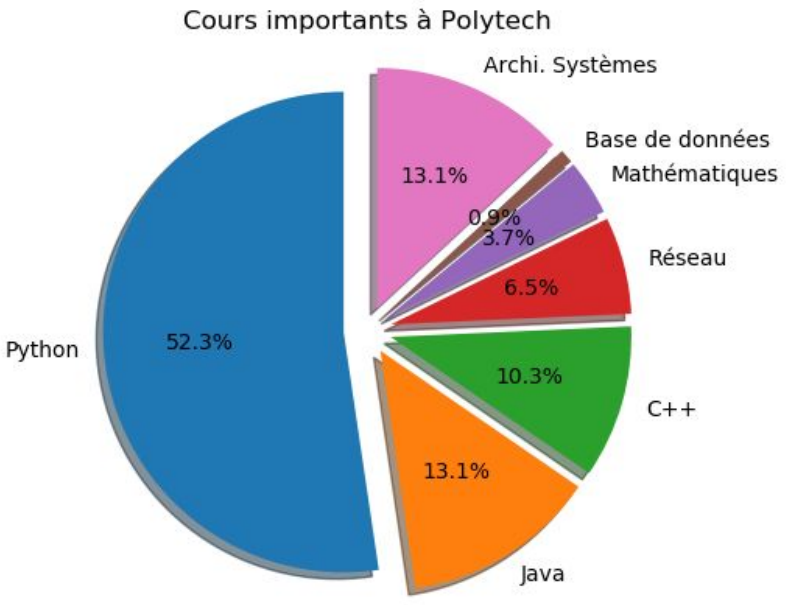
4. Camembert

Le graphe camembert s'appelle "Pie chart" en anglais, c'est donc la fonction `pie` que nous allons utiliser ici.

```
labels = 'Python', 'Java', 'C++', 'Réseau', 'Mathématiques', 'Base de
données', 'Archi. Systèmes'
sizes = [56, 14, 11, 7, 4, 1, 14]
explode = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)

fig1, ax1 = plt.subplots()

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a
circle.

plt.title("Cours importants à Polytech")
```

Ici nos valeurs dépassent les 100%, mais **Matplotlib** se débrouille pour créer un camembert qui va de 0 à 100% en adaptant nos données pour qu'elles soient correctement reflétées dans le graphe. Nous l'avons découvert en souhaitant expérimenter la gestion de cette erreur et c'est assez intéressant de le savoir.

La fonction `explode` permet de séparer les "parts".



5. Conclusion

Matplotlib à une utilisation assez instinctive et est facile à prendre en main. Cette bibliothèque permet de faire des graphes basiques mais il faut se souvenir que c'est aussi un outil puissant qui peut être très utile dans le domaine scientifique et permettre des réalisations bien plus complexes.

V. TP5 : Base de données

Ce TP a été le plus long de tous. Le code étant assez rébarbatif nous ne détaillerons pas tout dans le rapport, le lecteur pourra se référer aux sources pour plus de détails.

Dans un premier temps nous avons lu un fichier avec un parseur XML que nous avons écrit et qui entre tout dans une base de données à l'aide de commandes SQL. C'est très lourd car… car c'est du SQL tout simplement et donc ça nous produit énormément de ligne de code souvent assez similaires.

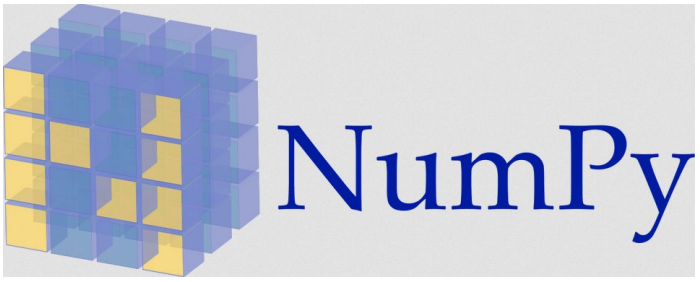
Ce parseur lit un fichier de données qui nous a posé problème au début car il faut trouver le bon encodage pour le lire. Une fois fait le reste va de soit.

Nous avons terminé le TP par quelques calculs sur la base de données en SQL. La fonction *exécute* est au final ce qu'il y a de plus important à retenir : on lui passe en paramètre une commande SQL valide et il l'exécute tout simplement.

```
cursor.execute("""SELECT nomRegion, populationTotale FROM regions""")
```

VI. TP6 : Numpy et Scipy

1. Numpy



Introduction

NumPy est le paquet fondamental du calcul scientifique avec Python. Il contient entre autres:

- un puissant objet tableau N-dimensionnel
- des fonctions sophistiquées
- des outils d'intégration de code C / C ++ et Fortran
- des utilisations d'algèbre linéaire: transformée de Fourier et capacités de nombres aléatoires

Outre ses utilisations scientifiques évidentes, NumPy peut également être utilisé comme un conteneur multidimensionnel efficace de données génériques. Des types de données arbitraires peuvent être définis. Cela permet à NumPy de s'intégrer de manière transparente et rapide à une grande variété de bases de données.

Travaux Pratiques

Pour commencer nous allons créer un tableau rempli aléatoirement de dimension 3 et de shape (4, 3, 2). Pour cela nous allons importer la bibliothèque *Numpy* puis nous allons l'utiliser sous le nom de ***np***. Pour prendre des valeur aléatoires nous utilisons la fonction `rand` de cette bibliothèque.

```
x = np.random.rand(4, 3, 2)
```

On obtient un tableau aléatoire de dimension 3 de cette forme:

```
Tableau :
 [[[0.51099439 0.02112441]
  [0.97932379 0.91819697]
  [0.23081341 0.23419679]]

 [[0.43366187 0.27378055]
  [0.05875391 0.42205515]
  [0.21108316 0.27617077]]

 [[0.08340248 0.30194098]
  [0.60137902 0.66831185]
  [0.29904179 0.15559798]]

 [[0.42549073 0.41588344]
  [0.12703182 0.10151253]
  [0.85869765 0.76286958]]]
```

Nous pouvons également obtenir diverses informations sur notre tableau en utilisant les fonctions de cette bibliothèque comme par exemple `dim`, `shape`, `size`, etc...

```
nd = np.ndim(x)
```

```
sh = np.shape(x)
```

=>

```
dimension : 3
shape:  (4, 3, 2)
size : 24
```

```
si = np.size(x)
```

Dans la suite du TP nous allons créer 2 matrices 3x3 initialisées avec les entiers de 0 à 8 pour la première et de 2 à 10 pour la seconde et en calculer le produit. Pour créer les matrice nous allons de nouveau utiliser Numpy et pour spécifier quels entiers nous voulons dedans nous pouvons utiliser `randint` ou `random_integers`.

```
m1 = np.random.randint(9, size=(3, 3))

m2 = np.random.random_integers(2, 10, size=(3, 3))
```

On obtient deux matrices qui remplissent nos conditions :

```
Matrice 1 :
  [[6 3 5]
 [1 6 8]
 [4 1 1]]
Matrice 2 :
  [[ 2 10 10]
 [ 9  4  9]
 [ 5  5 10]]
```

Nous allons maintenant pouvoir faire des calculs sur nos matrices comme par exemple leur produit, la transposé ou le déterminant. Toutes ces fonctions sont disponible dans la bibliothèque *Numpy* de la façon suivante :

```
prod = np.dot(m1, m2)

trans = m1.transpose()

detM1 = np.linalg.det(m1)
```

On obtient ainsi nos résultats :

```
Produit M1M2 :
  [[ 64  97 137]
 [ 96  74 144]
 [ 22  49  59]]
Transposé de la M1 :
  [[6 1 4]
 [3 6 1]
 [5 8 1]]
Determinant de M1 :
  -33.99999999999999
```

Enfin *Numpy* permet de réaliser des calculs plus complexes comme par exemple résoudre le systèmes d'équations ou bien récupérer les vecteurs propres d'un matrice :

```
solvM1M2 = np.linalg.solve(m1, m2)


valVectProprM1 = np.linalg.eig(m1)
```

Résultat :

```
Résolution du systeme M1M2 :
  [[ 0.47058824  1.23529412  1.82352941]
 [ 8.20588235 -1.14705882  7.23529412]
 [-5.08823529  1.20588235 -4.52941176]]
Vecteurs propres M1 :
  (array([11.21359372, -1.06379611,  2.85020239]), array([[-0.68947817, -0.16038301,  0.36046187],
       [-0.64321889, -0.72987033, -0.88240645],
       [-0.33300047,  0.66450477,  0.30236748]]))
```

2. Scipy

Introduction

SciPy est une bibliothèque Python gratuite et à code source ouvert utilisée pour le calcul scientifique et le calcul technique.

Elle contient des modules d'optimisation, d'algèbre linéaire, d'intégration, d'interpolation, de fonctions spéciales, de FFT, de traitement du signal et des images, de résolution d'EDO et d'autres tâches courantes en sciences et en ingénierie.

SciPy s'appuie sur *NumPy* et fait partie de la pile *NumPy*, qui inclut des outils tels que Matplotlib, pandas et SymPy.

Travaux Pratiques

Nous allons ici tenter d’approcher des points par une courbe, pour cela il nous faut un fichier contenant les coordonnées de nos points, les tracer et ensuite les approximer par une courbe.

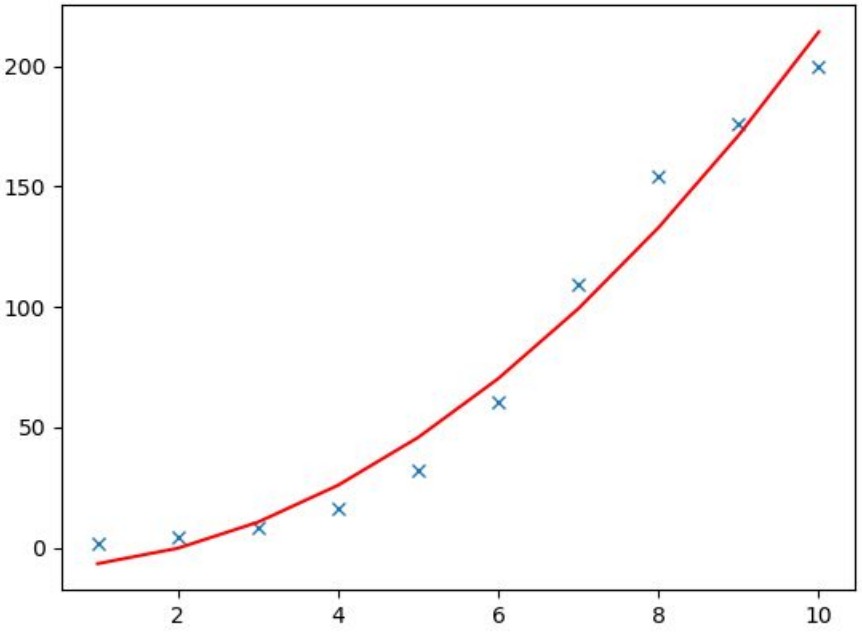
Notre fichier de données se présente de la façon suivante :

```
1.0  2.0
2.0  4.2
3.0  8.4
4.0  16.1
5.0  32.3
6.0  60.7
7.0  109.5
8.0  154.3
9.0  176.2
10.0 200.0
```

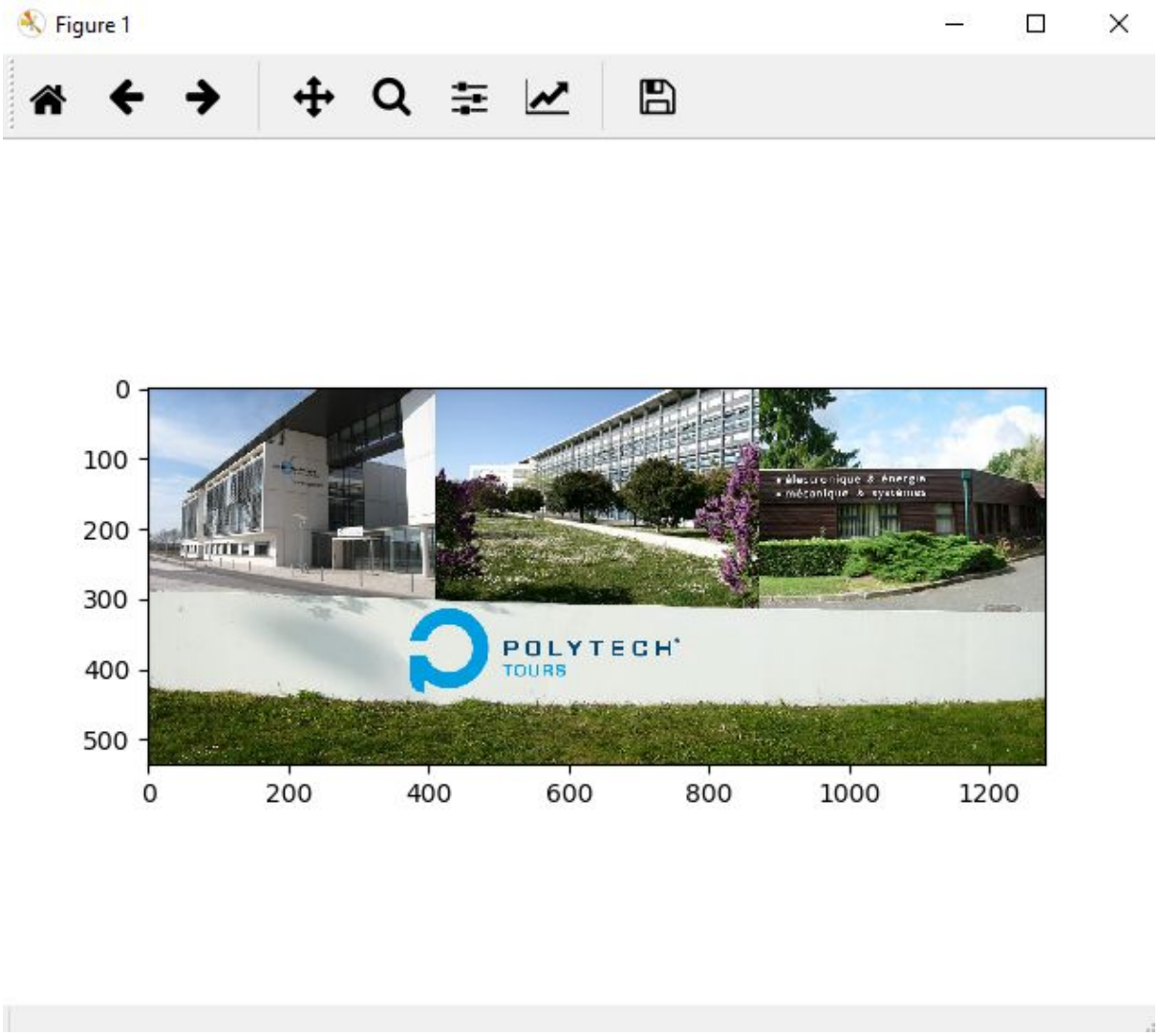
Nous allons le lire et remplir nos données x et y de la façon suivante :

```
data = plb.loadtxt('data.dat')

x = data[:, 0]

y = data[:, 1]
```

Afin d'approximer nos valeur avec une courbe nous allons faire appel à la fonction `curv_fit` appartenant à la librairie `scipy.optimize`. Pour afficher le résultats nous utilisons les fonctions `plot` et `show` de la librairie `matplotlib.pyplot`.



Enfin nous allons apprendre à afficher une image présente sur notre machine à l'aide de la fonction `imread` de *scipy*. Pour cela il suffit d'appeler la-dite fonction puis de lui indiquer le chemin de notre image. La fonction `imshow` de `matplotlib` va venir ici remplacer la fonction `plot` de l'exemple précédent puis nous utiliserons de la même façon la fonction `show` pour afficher notre résultat.



De plus, il est possible de réaliser diverses modifications de cette image en utilisant *scipy* avec par exemple la fonction `imresize` qui va nous permettre de redimensionner notre image. Pour cela nous procédons de la façon suivante:

```
img= misc.imread("E:\Polytech\Python\TP6\image.png")
img_resized= misc.imresize(img, 0.5)
plt.imshow(img_resized)

plt.show()
```

Dans cette exemple, `imresize` prends en paramètres notre image ainsi que la valeur 0.5. Cela indique que nous voulons diviser par 2 la taille de l'image, on obtient donc :

Notez la différence des échelles avec le résultat précédent

3. Conclusion

L'usage de ces 2 bibliothèques s'avère très utile pour réaliser des calculs et exploiter des résultats scientifique. De plus, elles sont toutes 2 très bien documentées sur le web ce qui les rend facile à utiliser. Dans ces exemples nous avons seulement effleuré leur capacités qui peuvent être bien plus poussées.

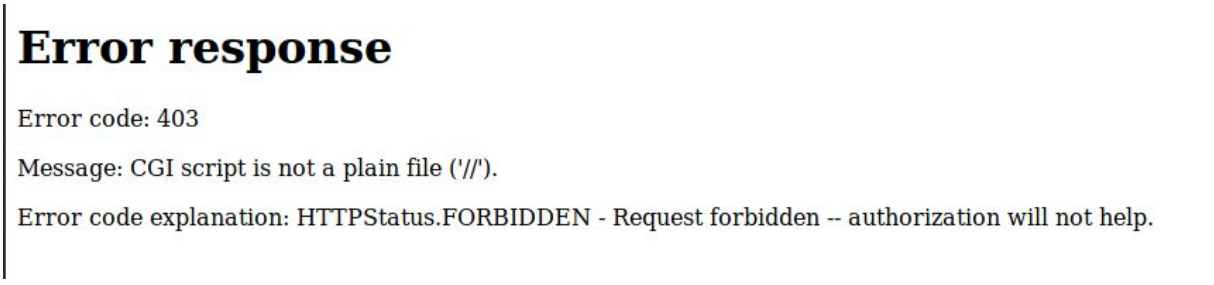
VII. TP7 : Serveur et page web

Lorsque nous avons fait ce TP nous avons rencontrés quelques problèmes. En effet, lors de la toute première question il est demandé de lancer un simple serveur en python qui sera à retravailler lors du reste du TP pour s'exercer. Seulement nous n'avons malheureusement jamais réussi à faire fonctionner ce dernier.

En effet, comme nous pouvons le voir sur les deux images suivantes nous avons une erreur CGI que nous n'avons pas su résoudre. Malgré toutes les aides que nous avons pu recevoir (camarades, professeur et Internet) et malgré nos différents tests (Windows et Linux) ce problème n'a jamais été résolu et nous avons fini par choisir de passer à la suite pour ne pas perdre trop de temps car nous avions déjà engendré beaucoup trop de retard.

```
/usr/bin/python3.7 /home/seimu/Projects/DI5S9/Python/TP7/server.py
Serveur actif sur le port : 8888
127.0.0.1 - - [17/Nov/2018 00:31:45] code 403, message CGI script is not a plain file ('//')
127.0.0.1 - - [17/Nov/2018 00:31:45] "GET / HTTP/1.1" 403 -
127.0.0.1 - - [17/Nov/2018 00:31:45] code 404, message No such CGI script ('//favicon.ico')
127.0.0.1 - - [17/Nov/2018 00:31:45] "GET /favicon.ico HTTP/1.1" 404 -
```

Erreur obtenue sur le programme Python.



Erreur obtenue sur le navigateur.

VIII. TP8 : Prog. asynchrone et fourmis

Multi-threading

Dans un premier temps nous avons implémenté comme demandé le multi-threading. C'est très simple, on fait passer le paramètre *Thread* dans l'implémentation de la classe, on l'appel dans le constructeur et on fait une méthode *run* qui sera la fonction appelée en multi-thread :

```
from threading import Thread


class MultiThreadingExample(Thread):
  def __init__(self):
      Thread.__init__(self)

  def run(self):
      n = 1E7
      while n > 0:
          n -= 1
```

Multi-processing

Encore une fois, c'est très simple. Pour faire du multi-processing nous avons juste à définir une méthode start et le tour est joué :

```
from multiprocessing import Process


class MultiProcessingExample:
  def calcul(self):
      n = 1E7
      while n > 0:
          n -= 1

  def start(self):
      p = Process(target=self.calcul)
      p.start()
      p.join()
```

IV. Conclusion

Pour terminer, nous pouvons voir que nous avons expérimenté de nombreux domaines avec Python.

Certes nous sommes loin d'une utilisation que pourrait avoir un expert, mais ces TPs auront eu le mérite de nous faire découvrir de nombreuses possibilité. Et surtout, désormais nous savons où chercher si nous voulons faire quelque chose de particulier en Python.

C'est un langage très simple et les codes à produire sont très rapides, il est parfait pour faire des scripts ou des tests.

Annexe

Voici le code de chacun des programmes.

TP1

main.py

```
from Editor import *
from EtudiantReader import *


# Question 1
# print("Bonjour tout le monde !")
#
# # Question 2
# editor = Editor()
# editor.menu()

# Question 5
student_list = EtudiantReader("fichetu.csv")
print(student_list)
```

Editor.py

```
class Editor:
  filename = ""

  def __init__(self) -> None:
    super().__init__()

  def menu(self):
    choice = 0
    while(choice != 9):
      while((choice < 1 or choice > 4) and choice != 9):
        print("*------------ Menu ------------*")
        print("1. Choisir un nom de fichier")
        print("2. Ajouter un texte")
        print("3. Afficher le fichier complet")
        print("4. Vider le fichier")
        print("9. Quitter le programme")
        choice = int(input("Choix : "))
```

```
        if(choice == 1):
            self.choose_filename()
        elif(choice == 2):
            self.add_text()
        elif(choice == 3):
            self.print_file()
        elif(choice == 4):
            self.empty_file()
        if(choice != 9):
            choice = 0

    def choose_filename(self):
        self.filename = input("Veuillez entrer le nom du fichier : ")

    def add_text(self):
        if(self.filename == ""):
            print("Merci d'entrer un nom de fichier.")
        else:
            text_to_add = input("Entrez le texte à ajouter : ")
            file = open(self.filename, 'a')
            file.write(text_to_add)
            file.close()

    def print_file(self):
        if(self.filename == ""):
            print("Merci d'entrer un nom de fichier.")
        else:
            with open(self.filename, 'r') as file:
                print((file.read()))

    def empty_file(self):
        if(self.filename == ""):
            print("Merci d'entrer un nom de fichier.")
        else:
            open(self.filename, 'w').close()
```

Etudiant.py

```
from datetime import date
from Date import *


class Etudiant:
    nom = ""
    prenom = ""
    address = ""
    age = 0
```

```
    date_naissance = Date(0, 0, 0)

    def __init__(self, nom, prenom, date_naissance):
        self.nom = nom
        self.prenom = prenom
        self.date_naissance = Date(date_naissance)
        self.addressmail()
        self.makeAge()

    def addressmail(self):
        self.address = self.nom.lower().replace(" ", "") + '.' + self.prenom.lower().replace(" ", "")
+ "@etu.univ-tours.fr"

    # TODO.
    def makeAge(self):
        today = date.today()
        self.age = today.year - self.date_naissance.annee
        if(self.date_naissance.mois == today.month):
            if(self.date_naissance.jour < today.day):
                self.age -= 1
        elif(self.date_naissance.mois < today.month):
                self.age -= 1

    def getNom(self):
        return self.nom

    def getPrenom(self):
        return self.prenom

    def getAddress(self):
        return self.address

    def getAge(self):
        return self.age

    def getDateNaissance(self):
        return self.date_naissance

    def __str__(self) -> str:
        return self.prenom[0] + self.prenom[1:].lower() + " " + self.nom[0] +
self.nom[1:].lower() + " a " + str(self.age) + " ans et son adresse mail est : " + self.address
```

EtudiantReader.py

```
from Etudiant import *
```

```
class EtudiantReader:
    etudiants = []

    def __init__(self, filename) -> None:
        self.readEtudiants(filename)

    def readEtudiants(self, filename):
        lines = open(filename, 'r').readlines()
        for line in lines:
            etu_info = line.split(";")
            self.etudiants.append(Etudiant(etu_info[0], etu_info[1], etu_info[2]))

    def __str__(self) -> str:
        output = ""
        for e in self.etudiants:
            output += str(e) + "\n"
        return output
```

Date.py

```
class Date:
    jour = 0
    mois = 0
    annee = 0

    def __init__(self, *args):
        if(len(args) == 3):
            self.jour = int(args[0])
            self.mois = int(args[1])
            self.annee = int(args[2])
        elif(len(args) == 1):
            date = args[0].split('\n')[0]
            self.jour = int(date.split('/')[0])
            self.mois = int(date.split('/')[1])
            self.annee = int(date.split('/')[2])
        else:
            raise Exception("Date not valid.")
#           print("TODO : exception.")

    def __eq__(self, o: object):
        if(self.jour == o.jour and self.mois == o.mois and self.annee == o.annee):
            return True
        return False

    def __lt__(self, other) -> bool:
        if(self.annee < other.annee):
            return True
```

```
        elif(self.annee == other.annee):
            if(self.mois < other.mois):
                return True
            elif(self.mois == other.mois):
                if(self.jour < other.jour):
                    return True
        return False

    def __str__(self):
        return str(self.jour) + '/' + str(self.mois) + '/' + str(self.annee)
```

TP2

main.py

```
from Calculatrice import *


calculatrice = Calculatrice()
```

calculatrice.py

```
#!/usr/bin/python
# -*-coding: utf-8 -*-

from tkinter import *


class Calculatrice:
    window = Tk()
    label = Label(window, text="0")

    calcul = ""


    def AC_function(self):
        self.calcul = ""
        self.label.config(text=0)

    def C_function(self):
        self.calcul = self.calcul[:-1]
        self.label.config(text=self.calcul)
```

```
    def number_function(self, number):
        self.calcul += str(number)
        self.label.config(text=self.calcul)

    def plus_function(self):
        self.calcul += "+"
        self.label.config(text=self.calcul)

    def minus_function(self):
        self.calcul += "-"
        self.label.config(text=self.calcul)

    def multiply_function(self):
        self.calcul += "*"
        self.label.config(text=self.calcul)

    def divide_function(self):
        self.calcul += "/"
        self.label.config(text=self.calcul)

    def equal_function(self):
        try:
            self.calcul = str(eval(self.calcul))
            self.label.config(text=self.calcul)
        except ZeroDivisionError:
            self.label.config(text="ERROR, division by zero impossible.")
            self.calcul = ""
        except SyntaxError:
            self.label.config(text="Syntax error.")
            self.calcul = ""


    def __init__(self):
        self.label.grid(row=0, column=0, columnspan=4)

        b = Button(self.window, text="AC", command=self.AC_function, width=16, height=2)
        b.grid(row=1, column=0, columnspan=2)

        b = Button(self.window, text="C", command=self.C_function, width=16, height=2)
        b.grid(row=1, column=2, columnspan=2)

        b = Button(self.window, text=7, command=lambda: self.number_function(7), width=6,
height=2)
        b.grid(row=2, column=0)

        b = Button(self.window, text=8, command=lambda: self.number_function(8), width=6,
height=2)
        b.grid(row=2, column=1)

        b = Button(self.window, text=9, command=lambda: self.number_function(9), width=6,
```

```
height=2)
        b.grid(row=2, column=2)

        b = Button(self.window, text=4, command=lambda: self.number_function(4), width=6,
height=2)
        b.grid(row=3, column=0)

        b = Button(self.window, text=5, command=lambda: self.number_function(5), width=6,
height=2)
        b.grid(row=3, column=1)

        b = Button(self.window, text=6, command=lambda: self.number_function(6), width=6,
height=2)
        b.grid(row=3, column=2)

        b = Button(self.window, text=1, command=lambda: self.number_function(1), width=6,
height=2)
        b.grid(row=4, column=0)

        b = Button(self.window, text=2, command=lambda: self.number_function(2), width=6,
height=2)
        b.grid(row=4, column=1)

        b = Button(self.window, text=3, command=lambda: self.number_function(3), width=6,
height=2)
        b.grid(row=4, column=2)

        b = Button(self.window, text=0, command=lambda: self.number_function(0), width=6,
height=2)
        b.grid(row=5, column=0)


        b = Button(self.window, text="+", command=self.plus_function, width=6, height=2)
        b.grid(row=2, column=3)

        b = Button(self.window, text="-", command=self.minus_function, width=6, height=2)
        b.grid(row=3, column=3)

        b = Button(self.window, text="*", command=self.multiply_function, width=6, height=2)
        b.grid(row=4, column=3)

        b = Button(self.window, text="/", command=self.divide_function, width=6, height=2)
        b.grid(row=5, column=3)

        b = Button(self.window, text="=", command=self.equal_function, width=16, height=2)
        b.grid(row=5, column=1, columnspan=2)

        self.window.mainloop()
```

TP3

main.py

```
from UserCrypto import *


user = UserCrypto("pass.txt")

if(user.is_connected):
    choice = -1
    while(choice != 0):
        print("Que souhaitez vous faire ?")
        print("[0] Quitter")
        print("[1] Chiffrer")
        print("[2] Déchiffrer")
        choice = int(input("Choix : "))
        print(choice)
        if(choice == 1):
            filepath = input("Indiquez le chemin de votre fichier : ")
            user.encryptFile(filepath)
            print("Fichier chiffré en AES256 disponible à l'adresse suivante : " + filepath +
".encrypted")
        elif(choice == 2):
            filepath = input("Indiquez le chemin de votre fichier : ")
            user.uncryptFile(filepath)
            print("Fichier déchiffré disponible à l'adresse suivante : " +
filepath.split(".encrypted")[0])
        elif(choice != 0):
            print("Your choice doesn't exist.")
```

UserCrypto.py

```
import hashlib
import os.path
import base64
import getpass
from Crypto.Cipher import AES


class UserCrypto:
    passfile = "pass.txt"
    username = "admin"
    password = ""
```

```
salt = "t'esbiensaléhein ?"
is_connected = False

def __init__(self, passfile):
    self.passfile = passfile
    if(self.userExists()):
        username = input("Username : ")
        password = getpass.getpass("Password : ")
        self.connect(username, password)
    else:
        username = input("Please, enter a new username : ")
        password = getpass.getpass("Password : ")
        password_confirmer = getpass.getpass("Please, confirm the password : ")
        self.subscribe(username, password, password_confirmer)

def userExists(self):
    if(os.path.exists(self.passfile)):
        return True
    else:
        return False

def subscribe(self, username, password, password_confirm):
    if(password == password_confirm):
        passfile = open(self.passfile, "w")
        salt_password = username + password + username + self.salt
        self.password = hashlib.sha512(salt_password.encode()).hexdigest()
        passfile.write(username + ":" + self.password + '\n')
        self.is_connected = True
        return self.is_connected

def connect(self, username, password):
    file = open(self.passfile, "r")
    for line in file:
        if(line.split(":")[0] == username):
            salt_password = username + password + username + self.salt
            self.password = hashlib.sha512(salt_password.encode()).hexdigest()
            if (line.split(":")[1][0:-1] == self.password):
                print("Connecté !")
                self.is_connected = True
                return self.is_connected
            else:
                print("Mauvais mot de passe...")
                self.is_connected = False
                return self.is_connected
    print("Cet utilisateur n'existe pas.")
    return False

def encryptFile(self, filepath):
    file = open(filepath, "rb")
    data = b""
```

```
        lines = file.readlines()
        for line in lines:
            data = data + line + b"\n"

        cipher = AES.new(base64.b64decode(self.password)[:16], AES.MODE_EAX)
        ciphertext, tag = cipher.encrypt_and_digest(data)

        file_out = open(filepath + ".encrypted", "wb")
        [file_out.write(x) for x in (cipher.nonce, tag, ciphertext)]

        file.close()
        file_out.close()

    def uncryptFile(self, filepath):
        file_in = open(filepath, "rb")
        file_out = open(filepath.split(".encrypted")[0], "wb")

        nonce, tag, ciphertext = [file_in.read(x) for x in (16, 16, -1)]
        cipher = AES.new(base64.b64decode(self.password)[:16], AES.MODE_EAX, nonce)

        data = cipher.decrypt_and_verify(ciphertext, tag)
        data = data.split(b"\n")
        [file_out.write(x) for x in data]

        file_in.close()
        file_out.close()
```

TP4

points.py

```
import matplotlib.pyplot as plt
import random



random_numbers_X = []
random_numbers_Y = []

for i in range(200):
    random_numbers_X.append(random.randint(0, 200))
    random_numbers_Y.append(random.randint(0, 200))

print(random_numbers_X)
print(random_numbers_Y)
```

```
# Print random numbers on a graph.
plt.plot(random_numbers_X, random_numbers_Y, 'ro', color='b', label='Point aléatoire')
plt.annotate('Oh ! Il y a une flèche !', xy=(100, 100), xytext=(120, 120),
arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))
plt.xlabel('Valeurs aleatoire x')
plt.ylabel('Valeurs aleatoires y')
plt.title("Exemple de nuage de points")
plt.legend()
plt.show()
```

camembert.py

```
import matplotlib.pyplot as plt

labels = 'Python', 'Java', 'C++', 'Réseau', 'Mathématiques', 'Base de données', 'Archi.
Systèmes'
sizes = [56, 14, 11, 7, 4, 1, 14]
explode = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)

fig1, ax1 = plt.subplots()

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True,
startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title("Cours importants à Polytech")
plt.show()
```

histogram.py

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


# Histogram.
x = np.arange(2)
plt.bar(x, height=[10, 90])
plt.xticks(x, ['Hommes', 'Femmes'])

plt.ylabel('Pourcentage de personnes qui font le ménage')
plt.xlabel('Genres')
plt.title("Etude du partage des tâches ménagères")

plt.show()
```

TP5

main.py

```
from dbWorker import *
from XMLParser import *
import os


# Question 1
#########################################################################
dbWorker = dbWorker()
# TODO : mettre une condition pour vérifier si la BDD existe déjà et décommenter les lignes suivantes en conséquence.
if(not os.path.exists('tp5.db')):
   dbWorker.readCommunes("bdd/communes.csv")
   dbWorker.readDepartements("bdd/departements.csv")
   dbWorker.readRegions("bdd/regions.csv")
print("Question 1 : database up !")


# Init
#########################################################################################
db = sqlite3.connect('tp5.db')
cursor = db.cursor()


# Question 2
#########################################################################
print("Question 2, population totale des communes :")
cursor.execute("""SELECT nomCommune, populationTotale FROM communes""")
for commune in cursor.fetchall():
   print(commune)

print("Question 2, population totale des régions :")
cursor.execute("""SELECT nomRegion, populationTotale FROM regions""")
for region in cursor.fetchall():
   print("Dans la région \"" + region[0] + "\" il y a " + str(region[1]) + " habitants.")


# Question 3
#########################################################################
print("Question 3 :")
cursor.execute( """
            SELECT communes.nomCommune,
GROUP_CONCAT(departements.codeDepartement)
            FROM communes
            INNER JOIN departements ON communes.codeDepartement =
```

```
departements.codeDepartement
            GROUP BY communes.nomCommune
            HAVING COUNT(*) > 1
            ORDER BY COUNT(*)
            """)
for doublon in cursor.fetchall():
  print(doublon[0] + " existe dans les départements suivants : " + doublon[1])


# Question 4
################################################################################
print("Question 4 :")
dbWorker.writeIntoXML('db.xml')
print("Exportation to XML done.")
```

XMLParser.py

```
import sqlite3


def XMLWrite(filepath, title, tabNames, tabValues):
  file = open(filepath, 'a')
  file.write("<title>" + title + "</title>" + "\n")
  for value in tabValues:
    for i in range(len(tabNames)):
      file.write("    <" + tabNames[i] + ">" + str(value[i]) + "</" + tabNames[i] + ">" + "\n")
    file.write("\n")

# IMPORTANT : doesn't work, don't use it.
def XMLRead(filepath):
  file = open(filepath, 'r')
  number_of_titles = 0
  number_of_tabs = 0
  number_of_values = 0
  titles = []
  values = []

  # Read XML.
  for line in file.readlines():
    if(line.startswith("<title>")):
      titles.append(line.split("<title>")[1].split("</title>")[0])
      values.append([])
      number_of_titles = number_of_titles + 1
      number_of_tabs = 0
    elif(line.startswith("    <")):
      values[number_of_titles - 1].append([[]])
      values[number_of_titles - 1][number_of_tabs].append(line.split(">")[1].split("<")[0])
```

```
        number_of_tabs = number_of_tabs + 1

# Insert into DB.
db = sqlite3.connect('tp5.db.fromXML')
cursor = db.cursor()

# Add communes.
cursor.execute("""
CREATE TABLE IF NOT EXISTS communes(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    codeRegion INTEGER,
    nomRegion VARCHAR,
    codeDepartement VARCHAR,
    codeArrondissement INTEGER,
    codeCanton INTEGER,
    codeCommune INTEGER,
    nomCommune VARCHAR,
    populationMunicipale INTEGER,
    populationCompteeAPart INTEGER,
    populationTotale INTEGER
)
""")
db.commit()
for i in range(len(values[0])):
    data = {
        "codeRegion": int(values[0][i][0]),
        "nomRegion": values[0][i][1],
        "codeDepartement": values[0][i][2],
        "codeArrondissement": int(values[0][i][3]),
        "codeCanton": int(values[0][i][4]),
        "codeCommune": int(values[0][i][5]),
        "nomCommune": values[0][i][6],
        "populationMunicipale": int(values[0][i][7]),
        "populationCompteeAPart": int(values[0][i][8]),
        "populationTotale": int(values[0][i][9])
    }
    cursor.execute("""
    INSERT INTO communes(   codeRegion,
                            nomRegion,
                            codeDepartement,
                            codeArrondissement,
                            codeCanton,
                            codeCommune,
                            nomCommune,
                            populationMunicipale,
                            populationCompteeAPart,
                            populationTotale)
    VALUES( :codeRegion,
            :nomRegion,
            :codeDepartement,
```

```
            :codeArrondissement,
            :codeCanton,
            :codeCommune,
            :nomCommune,
            :populationMunicipale,
            :populationCompteeAPart,
            :populationTotale)
    """, data)
db.commit()

# Add departements.
cursor.execute("""
CREATE TABLE IF NOT EXISTS departements(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    codeRegion INTEGER,
    nomRegion VARCHAR,
    codeDepartement VARCHAR,
    nomDepartement VARCHAR,
    nombreArrondissements INTEGER,
    nombreCantons INTEGER,
    nombreCommunes INTEGER,
    populationMunicipale INTEGER,
    populationTotale INTEGER
)
""")
db.commit()
for i in range(len(values[1])):
    data = {
        "codeRegion": int(values[1][i][0]),
        "nomRegion": values[1][i][1],
        "codeDepartement": values[1][i][2],
        "nomDepartement": values[1][i][3],
        "nombreArrondissements": int(values[1][i][4]),
        "nombreCantons": int(values[1][i][5]),
        "nombreCommunes": int(values[1][i][6]),
        "populationMunicipale": int(values[1][i][7]),
        "populationTotale": int(values[1][i][8]),
    }
    cursor.execute("""
    INSERT INTO departements(   codeRegion,
                                nomRegion,
                                codeDepartement,
                                nomDepartement,
                                nombreArrondissements,
                                nombreCantons,
                                nombreCommunes,
                                populationMunicipale,
                                populationTotale)
    VALUES( :codeRegion,
            :nomRegion,
```

```
            :codeDepartement,
            :nomDepartement,
            :nombreArrondissements,
            :nombreCantons,
            :nombreCommunes,
            :populationMunicipale,
            :populationTotale)
    """, data)
db.commit()

# Add Departements.
cursor.execute("""
CREATE TABLE IF NOT EXISTS regions(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    codeRegion INTEGER,
    nomRegion VARCHAR,
    nombreArrondissements INTEGER,
    nombreCantons INTEGER,
    nombreCommunes INTEGER,
    populationMunicipale INTEGER,
    populationTotale INTEGER
)
""")
db.commit()
for i in range(values[2]):
    data = {
        "codevalues[2][i]": int(values[2][i][0]),
        "nomRegion": values[2][i][1],
        "nombreArrondissements": int(values[2][i][2]),
        "nombreCantons": int(values[2][i][3]),
        "nombreCommunes": int(values[2][i][4]),
        "populationMunicipale": int(values[2][i][5]),
        "populationTotale": int(values[2][i][6])
    }
    cursor.execute("""
    INSERT INTO regions (   codeRegion,
                            nomRegion,
                            nombreArrondissements,
                            nombreCantons,
                            nombreCommunes,
                            populationMunicipale,
                            populationTotale)
    VALUES( :codeRegion,
            :nomRegion,
            :nombreArrondissements,
            :nombreCantons,
            :nombreCommunes,
            :populationMunicipale,
            :populationTotale)
    """, data)
```

```
db.commit()
```

dbWorker.py

```
import sqlite3
import os
from XMLParser import *


class dbWorker:
    db = ""

    def __init__(self):
        self.db = sqlite3.connect('tp5.db')

    def readCommunes(self, filename):
        # Create table.
        cursor = self.db.cursor()
        cursor.execute("""
        CREATE TABLE IF NOT EXISTS communes(
            id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
            codeRegion INTEGER,
            nomRegion VARCHAR,
            codeDepartement VARCHAR,
            codeArrondissement INTEGER,
            codeCanton INTEGER,
            codeCommune INTEGER,
            nomCommune VARCHAR,
            populationMunicipale INTEGER,
            populationCompteeAPart INTEGER,
            populationTotale INTEGER
        )
        """)
        self.db.commit()
        # Read CSV and insert data into the DB.
        with open(filename, 'rb') as file:
            for commune in file.readlines():
                commune = commune.decode("Windows-1252")
                commune = commune.split(";")
                try:
                    if(int(commune[0]) >= 0 and int(commune[0]) <= 99):
                        for i in [2, 3, 4, 5, 7, 8, 9]:
                            if " " in commune[i]:
                                commune[i] = commune[i].replace(' ', '')
                        data = {
                            "codeRegion": int(commune[0]),
                            "nomRegion": commune[1],
```

```
                    "codeDepartement": commune[2],
                    "codeArrondissement": int(commune[3]),
                    "codeCanton": int(commune[4]),
                    "codeCommune": int(commune[5]),
                    "nomCommune": commune[6],
                    "populationMunicipale": int(commune[7]),
                    "populationCompteeAPart": int(commune[8]),
                    "populationTotale": int(commune[9])
                }
                cursor.execute("""
                INSERT INTO communes(   codeRegion,
                                nomRegion,
                                codeDepartement,
                                codeArrondissement,
                                codeCanton,
                                codeCommune,
                                nomCommune,
                                populationMunicipale,
                                populationCompteeAPart,
                                populationTotale)
                VALUES( :codeRegion,
                        :nomRegion,
                        :codeDepartement,
                        :codeArrondissement,
                        :codeCanton,
                        :codeCommune,
                        :nomCommune,
                        :populationMunicipale,
                        :populationCompteeAPart,
                        :populationTotale)
                """, data)
            except Exception as e:
                pass
        self.db.commit()

    def readDepartements(self, filename):
        # Create table.
        cursor = self.db.cursor()
        cursor.execute("""
        CREATE TABLE IF NOT EXISTS departements(
            id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
            codeRegion INTEGER,
            nomRegion VARCHAR,
            codeDepartement VARCHAR,
            nomDepartement VARCHAR,
            nombreArrondissements INTEGER,
            nombreCantons INTEGER,
            nombreCommunes INTEGER,
            populationMunicipale INTEGER,
            populationTotale INTEGER
```

```
    )
    """)
    self.db.commit()
    # Read CSV and insert data into the DB.
    with open(filename, 'rb') as file:
        for departement in file.readlines():
            departement = departement.decode("Windows-1252")
            departement = departement.split(";")
            try:
                if(int(departement[0]) >= 0 and int(departement[0]) <= 99):
                    for i in [0, 4, 5, 6, 7, 8, 9]:
                        if " " in departement[i]:
                            departement[i] = departement[i].replace(' ', '')
                    data = {
                        "codeRegion": int(departement[0]),
                        "nomRegion": departement[1],
                        "codeDepartement": departement[2],
                        "nomDepartement": departement[3],
                        "nombreArrondissements": int(departement[4]),
                        "nombreCantons": int(departement[5]),
                        "nombreCommunes": int(departement[6]),
                        "populationMunicipale": int(departement[7]),
                        "populationTotale": int(departement[8]),
                    }
                    cursor.execute("""
                    INSERT INTO departements(   codeRegion,
                                                nomRegion,
                                                codeDepartement,
                                                nomDepartement,
                                                nombreArrondissements,
                                                nombreCantons,
                                                nombreCommunes,
                                                populationMunicipale,
                                                populationTotale)
                    VALUES( :codeRegion,
                            :nomRegion,
                            :codeDepartement,
                            :nomDepartement,
                            :nombreArrondissements,
                            :nombreCantons,
                            :nombreCommunes,
                            :populationMunicipale,
                            :populationTotale)
                    """, data)
            except Exception as e:
                pass
        self.db.commit()

def readRegions(self, filename):
    # Create table.
```

```
        cursor = self.db.cursor()
        cursor.execute("""
        CREATE TABLE IF NOT EXISTS regions(
            id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
            codeRegion INTEGER,
            nomRegion VARCHAR,
            nombreArrondissements INTEGER,
            nombreCantons INTEGER,
            nombreCommunes INTEGER,
            populationMunicipale INTEGER,
            populationTotale INTEGER
        )
        """)
        self.db.commit()
        # Read CSV and insert data into the DB.
        with open(filename, 'rb') as file:
            for region in file.readlines():
                region = region.decode("Windows-1252")
                region = region.split(";")
                try:
                    if(int(region[0]) >= 0 and int(region[0]) <= 99):
                        for i in [0, 2, 3, 4, 5, 6]:
                            if " " in region[i]:
                                region[i] = region[i].replace(' ', '')
                        data = {
                            "codeRegion": int(region[0]),
                            "nomRegion": region[1],
                            "nombreArrondissements": int(region[2]),
                            "nombreCantons": int(region[3]),
                            "nombreCommunes": int(region[4]),
                            "populationMunicipale": int(region[5]),
                            "populationTotale": int(region[6])
                        }
                        cursor.execute("""
                        INSERT INTO regions (   codeRegion,
                                                nomRegion,
                                                nombreArrondissements,
                                                nombreCantons,
                                                nombreCommunes,
                                                populationMunicipale,
                                                populationTotale)
                        VALUES( :codeRegion,
                                :nomRegion,
                                :nombreArrondissements,
                                :nombreCantons,
                                :nombreCommunes,
                                :populationMunicipale,
                                :populationTotale)
                        """, data)
                except Exception as e:
```

```
                pass
        self.db.commit()

    def writeIntoXML(self, filepath):
        cursor = self.db.cursor()

        if(os.path.exists(filepath)):
            os.remove(filepath)

        # Write communes.
        communesTabs = ['id', 'codeRegion', 'nomRegion', 'codeDepartement',
'codeArrondissement', 'codeCanton',
                        'codeCommune', 'nomCommune', 'populationMunicipale',
'populationCompteeAPart',
                        'populationTotale']
        cursor.execute("""SELECT * FROM communes""")
        communesValues = cursor.fetchall()
        XMLWrite(filepath, "communes", communesTabs, communesValues)

        # Write departments.
        departementsTabs = ['id', 'codeRegion', 'nomRegion', 'codeDepartement',
'nomDepartement',
                            'nombreArrondissements', 'nombreCantons', 'nombreCommunes',
'populationMunicipale',
                            'populationTotale']
        cursor.execute("""SELECT * FROM departements""")
        departementsValues = cursor.fetchall()
        XMLWrite(filepath, "departements", departementsTabs, departementsValues)

        # Write regions.
        regionsTabs = ['id', 'codeRegion', 'nomRegion', 'nombreArrondissements',
'nombreCantons', 'nombreCommunes',
                       'populationMunicipale', 'populationTotale']
        cursor.execute("""SELECT * FROM regions""")
        regionsValues = cursor.fetchall()
        XMLWrite(filepath, "regions", regionsTabs, regionsValues)
```

TP6

NympyPart.py

```
import numpy as np

# python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose
```

```
def tableau():
    x = np.random.rand(4, 3, 2)

    nd = np.ndim(x)
    sh = np.shape(x)
    si = np.size(x)
    # dt = np.dtype(x)

    print("\n", "dimension :", nd, "\n", "shape: ", sh, "\n", "size :", si, "\n")
    print("Tableau :\n", x)


tableau()

m1 = np.array([[0., 1., 2.],
               [3., 4., 5.],
               [6., 7., 8.]])

m2 = np.array([[2., 3., 4.],
               [5., 6., 7.],
               [8., 9., 10.]])

m3 = np.array([[2., 3.],
               [4., 5.]])

m4 = np.array([[1., 1.],
               [0., 1.]])

m5 = np.array([[1.], [1.]])

print("Produit M1M2 (avec *, multiplie chaque champ du tableau) : \n ", m1 * m2)
print("Produit M1M2 (avec dot, multiplie les deux matrices) : \n ", np.dot(m1, m2))
print("Transposée de la matrice M1 : \n ", m1.transpose())
print("Determinant de M3 : \n ", np.linalg.det(m3))
print("Inverse de M3 : \n ", np.linalg.inv(m3))
print("Résolution du systeme M1M2 : \n ", np.linalg.solve(m4, m5))
print("Vecteurs propres de M1 : \n ", np.linalg.eig(m1))
```

ScipyPart.py

```
import matplotlib.pyplot as plt
import scipy as sy
import pylab as plb
from scipy import misc
from scipy.misc import imresize, imshow, imread
from scipy.optimize import curve_fit
```

```
data = plb.loadtxt('data.dat')
x = data[:, 0]
y = data[:, 1]


def func(x, a, b, c):
    return a * x ** b + c

def approxPoints():
    p0 = sy.array([1, 1, 1])
    coeffs, matcov = curve_fit(func, x, y, p0)

    yaj = func(x, coeffs[0], coeffs[1], coeffs[2])
    print(coeffs)
    print(matcov)

    plt.plot(x, y, 'x', x, yaj, 'r-')
    plt.show()

# img= misc.imread("E:\Polytech\Python\TP6\image.png")
# img_resized= misc.imresize(monitor, 0.5)
# plt.imshow(img_resized)
# plt.imshow(img)
# plt.show()

approxPoints()
```

TP7

server.py

```
#!/usr/bin/python3

import http.server


PORT = 8888
server_address = ("", PORT)

server = http.server.HTTPServer
handler = http.server.CGIHTTPRequestHandler
handler.cgi_directories = ["/"]
print("Serveur actif sur le port :", PORT)

httpd = server(server_address, handler)
httpd.serve_forever()
```

index.py

```
#!/usr/bin/python3

import cgi


form = cgi.FieldStorage()
print("Content-type: text/html; charset=utf-8\n")

print(form.getvalue("name"))

html = """
  <!DOCTYPE html>
    <head>
      <title>Mon programme</title>
    </head>
    <body>
      <form action="/index.py" method="post">
        <input type="text" name="name" value="Votre nom" />
        <input type="submit" name="send" value="Envoyer information au serveur">
      </form>
    </body>
  </html>
"""

print(html)
```

TP8

main.py

```
from MultiThreadingExample import *
from MultiProcessingExample import *
from ParametersReader import *
from Fourmi import *


print("Start multi threading.")
thread1 = MultiThreadingExample()
thread1.start()

print("Start multi processing.")
```

```
thread2 = MultiProcessingExample()
thread2.start()

# print("Start fourmi.")
# reader = ParametersReader("params.xml")
# canvasHeight = 0
# canvasWeight = 0
# numberFourmi = 0
# fourmis = []
#
# for i in range(numberFourmi):
#     fourmis.append(canvas, Fourmi(reader.readNext("colorR"),
reader.readNext("colorV"), reader.readNext("colorB"), reader.readNext("followR"),
reader.readNext("followV"), reader.readNext("followB"), reader.readNext("probaG"),
reader.readNext("probaD"), reader.readNext("probaT"), reader.readNext("doorDd"),
reader.readNext("ps"), reader.readNext("numberIt"), reader.readNext("numberItDone")))
#
# for i in range(numberFourmi):
#     fourmis[i].start()
```

MultiThreadingExample.py

```
from threading import Thread


class MultiThreadingExample(Thread):
  def __init__(self):
      Thread.__init__(self)

  def run(self):
      n = 1E7
      while n > 0:
          n -= 1
```

MultiProcessingExample.py

```
from multiprocessing import Process


class MultiProcessingExample:
  def calcul(self):
      n = 1E7
      while n > 0:
          n -= 1
```

```
    def start(self):
        p = Process(target=self.calcul)
        p.start()
        p.join()
```

Fourmi.py

```
from multiprocessing import Process


class Fourmi:
    canvas = 0
    colorR = 0 # 0-255
    colorV = 0 # 0-255
    colorB = 0 # 0-255
    followR = 0
    followV = 0
    followB = 0
    probaG = 0 # G+D+T = 1
    probaD = 0
    probaT = 0
    doorDd = 0
    Ps = 0
    numberIt = 0
    numberItDone = 0

    def __init__(self, canvas, colorR, colorV, colorB, followR, followV, followB, probaG,
probaD, probaT, doorDd, Ps, numberIt, numberItDone):
        self.canvas = canvas
        self.colorR = colorR
        self.colorV = colorV
        self.colorB = colorB
        self.followR = followR
        self.followV = followV
        self.followB = followB
        self.probaG = probaG
        self.probaD = probaD
        self.probaT = probaT
        self.doorDd = doorDd
        self.ps = Ps
        self.numberIt = numberIt
        self.numberItDone = numberItDone

    def step(self):
        self.numberItDone += 1
```

```
    def start(self):
        p = Process(target=self.step)
        p.start()
        p.join()
```

ParametersReader.py

```
class ParametersReader:
    file = ""

    def __init__(self, filepath):
        self.file = open(filepath, 'r')

    def readNext(self, value):
        print("TODO.")
```