

Python TP

PENG Hanyuan & YAN Wenli

Les codes du TP sont fournis dans le fichier [Code_TP.zip](#) . (Python3.6)

TP1 - Fichiers

Ce sujet nous permet d'utiliser:

- Les fichiers: Lecture et l'écriture d'un fichier,
- Classe: Surcharge des méthodes, getter, setter.

Fichier

Quand nous voulons lire un fichier text (ou json, xml etc...), c'est mieux d'indiquer le codage de fichier pour que les textes affiche correctement.

```
def func():
    print(' Bonjour le monde !')
    print('\t1. Choisir un nom de fichier', '\t2. Ajouter un texte', '\t3. Afficher le
fichier complet', '\t4. Vider le fichier', '\t5. Quitter', sep='\n')

    filename = ''
    while True:
        inp = input('Votre choix: ')
        if inp == '1':
            filename = input('Nom de fichier: ')
            print('Nom du fichier:', filename)
        elif inp == '2':
            if not filename:
                print('Choisir un nom de fichier!')
            else:
                print('Nom du fichier: ', filename)
                with open(filename, "a", encoding="utf-8") as fic:
                    text = input("Votre Texte: ")
                    fic.write(text)
        elif inp == '3':
            if not filename:
                print('Choisir un nom de fichier!')
            else:
                print('Nom du fichier: ', filename)
                with open(filename, "r", encoding="utf-8") as fic:
                    print(fic.read(), end='\n')
        elif inp == '4':
            with open(filename, "w+") as fic:
                fic.truncate()
        elif inp == '5':
            break
        else:
            raise Exception("Error input, programme va quitter")

if __name__ == "__main__":
```

```
try:  
    func()  
except Exception as e:  
    print(e)
```

Résultat:

Bonjour le monde !

1. Choisir un nom de fichier
2. Ajouter un texte
3. Afficher le fichier complet
4. Vider le fichier
5. Quitter

Votre choix: 1

Nom de fichier: test

Nom du fichier: test

Votre choix: 2

Nom du fichier: test

Votre Texte: add text

Votre choix: 3

Nom du fichier: test

dsfwrgfshetadd text

Votre choix: ■

Classe

Surcharge des méthodes

```
class Date:  
    def __init__(self,date):  
        self.date = date  
    def __eq__(self, other):  
        if other.date == self.date:  
            return True  
        else:
```

```

        return False
def __lt__(self,other):
    if self.date < other.date:
        return True
    else:
        return False

date = Date(1.2)
date2 = Date(1.3)

print(date > date2)

```

Résultat:

```
→ TP1python3 TP1_Date.py
False
```

Des point importants

Il faut utiliser le nom particulier pour surcharger les opérateur (+, =, <, etc)

Attributs et Getter, Setter

Des codes de cette partie sont dans le [Code_TP.zip](#). Fichier [tp1_etudiant.py](#)

Résultat:

```
→ TP1 python3 tp1_etudiant.py
nom: ARCHAT
prenom: MATHIEU
age: 24
-----
nom: AZOUZ
prenom: OUHCHIA
age: 25
-----
nom: BAI
prenom: SHUO
age: 26
-----
```

TP2 - Tkinter

2.1 Introduction simple

Tkinter (de l'anglais Tool kit interface) est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques. Tkinter est intégré au paquet d'installation de python pour gérer rapidement des interfaces graphiques simples.

Tkinter offre une variété de composant d'interface graphique tels que [Boutons](#), [Label](#), [Frame](#), [Canvas](#), [Text](#), [Menu](#), [Message](#), [Scrollbar](#),etc.

Il possède des attributs standards comme: [Dimension](#), [Color](#), [Font](#), [Cusor](#), [Anchor](#), etc.

Le composant d'interface graphique de Tkinter a une méthode de gestion d'état géométrique spécifique qui gère l'organisation complète de la zone de contrôle. Voici les trois classes de gestion de géométrie exposée par Tkinter: [package\(\)](#), [grid\(\)](#), [place\(\)](#).

2.2 Travail du TP

Une calculatrice avec deux mode (standard et scientifique) réalisée par la bibliothèque `Math` et `Tkinter`. Ici, on a utilisé `grid()`:

```
def calBasique():
    global bg
    bg.title('Calculatrice')
    for btn in list_button_sci:
        btn.destroy()
    # bg.update_idletasks()
    # bg.geometry(400*200)
    global entry
    entry = Entry(justify='right')
    entry.grid(row=0, column=0, columnspan=5, sticky=N+W+S+E)

    button_c = Button(text='C', relief=RIDGE, width=5, command =lambda :
    clear(entry)).grid(row=4, column=1)
    button0 = Button(text='0', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '0')).grid(row=4, column=2)
    button_ac = Button(text='AC', relief=RIDGE, width=5, command =lambda :
    all_clear(entry)).grid(row=4, column=3)

    button1 = Button(text='1', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '1')).grid(row=3, column=1)
    button2 = Button(text='2', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '2')).grid(row=3, column=2)
    button3 = Button(text='3', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '3')).grid(row=3, column=3)

    button4 = Button(text='4', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '4')).grid(row=2, column=1)
    button5 = Button(text='5', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '5')).grid(row=2, column=2)
    button6 = Button(text='6', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '6')).grid(row=2, column=3)

    button7 = Button(text='7', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '7')).grid(row=1, column=1)
    button8 = Button(text='8', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '8')).grid(row=1, column=2)
    button9 = Button(text='9', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '9')).grid(row=1, column=3)

    button_plus = Button(text='+', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '+')).grid(row=1, column=4)
    button_minus = Button(text='-', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '-')).grid(row=2, column=4)
    button_multi = Button(text='*', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '*')).grid(row=3, column=4)
    button_div = Button(text='/', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '/')).grid(row=4, column=4)

    button_point = Button(text='.', relief=RIDGE, width=5, command =lambda :
    get_input(entry, '.')).grid(row=5, column=2)
    button_equ = Button(text='=', relief=RIDGE, width=5, command =lambda :
    cal(entry)).grid(row=5, column=3)
```

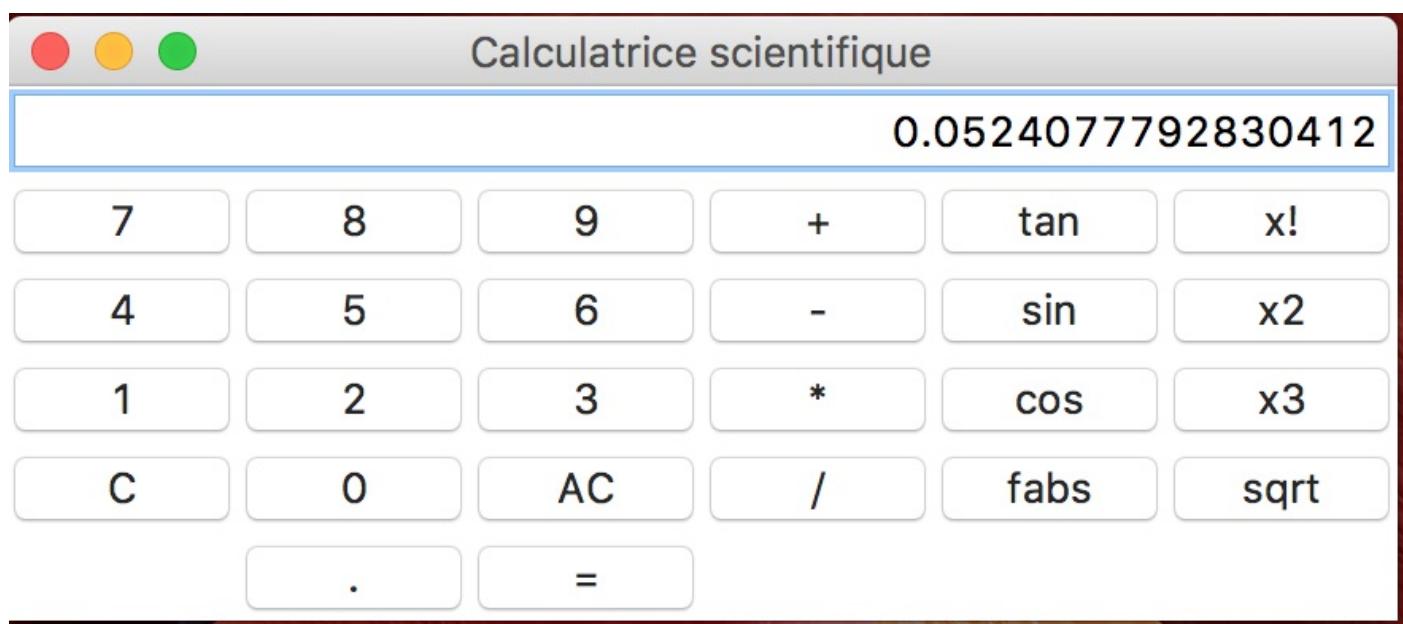
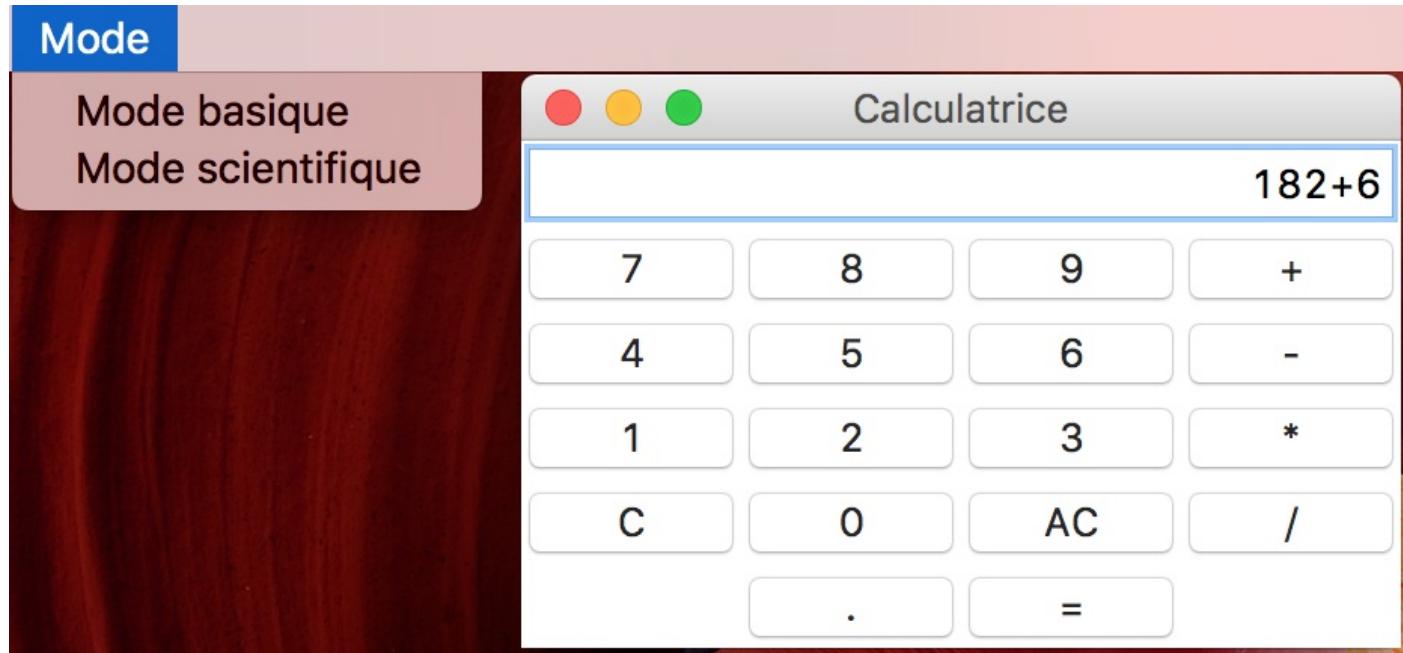
Pour chaque opération, on a défini sa fonction en utilisant des méthodes de `Math`, par exemple `calSin` (C'est pareil pour les autres opérations):

```

def calSin(entry):
    input = entry.get().strip()
    resultSin = math.sin((int(input)/180)* math.pi)
    all_clear(entry)
    entry.insert(END, resultSin)

```

Les résultats:



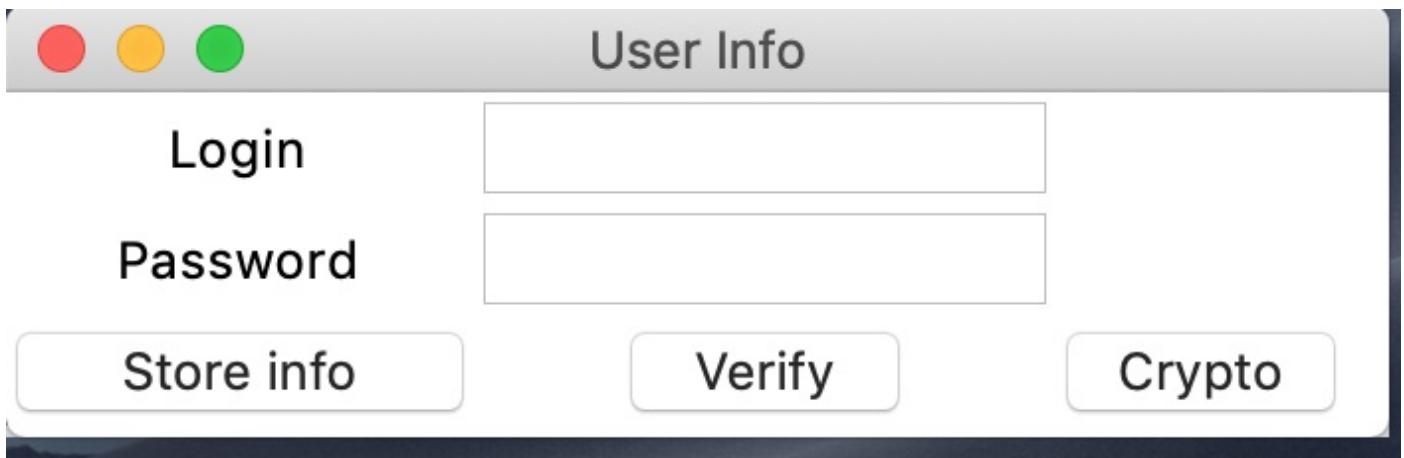
TP3 - Exceptions et chiffrement

Exceptions et assertion

Pour la partie de exceptions et assertion, nous les avons ajouté dans les tps précédents.

Chiffrement

Nous avons fait une interface ci-dessous avec Tkinter pour réaliser cette partie.



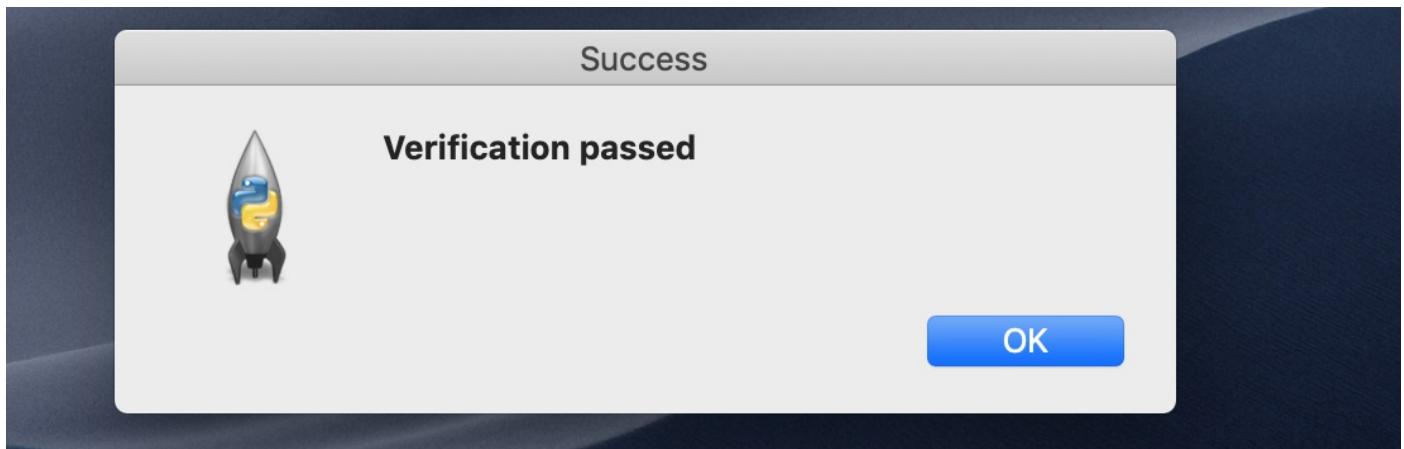
Après avoir saisi le login et mot de passe, vous pouvez cliquer `Store info` pour stocker les informations dans un fichier `userinfo.json`.

```
[  
] {  
    "login": "test",  
    "password": "5b0aee208df5d8e959ee99bf9f547251762f5ffd8f5aaa63fae0af1a6b77fdc7"  
},  
] {  
    "login": "tt",  
    "password": "1cd109e5820dfb09085ca6a938e9096aae5a22fe9675e8e6e288425e326e59ee"  
},  
] {  
    "login": "login",  
    "password": "9ad5ba25f80e18b14274d4a414938e30a23652342b3cb3891a3ffb0a7bd95bf9"  
},  
] {  
    "login": "aaa",  
    "password": "4415cf42f8cc23aa227f4a5cea0392407d267207135fc75d292ae26ffb739d85"  
}]
```

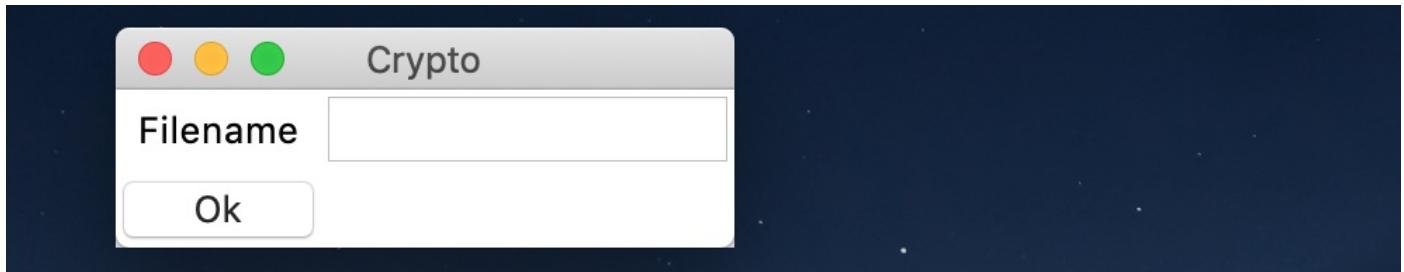
Les mots de passe sont chiffrés par l'algorithme `sha256` en utilisant le `'salt'`.

```
password = salt + password + login  
mdp = hashlib.sha256(password.encode()).hexdigest()
```

Ensuite vous pouvez cliquer `verify` pour vérifier les informations. Il y aura un message box qui indique si le résultat est correct.



Vous pouvez aussi cliquer `crypto`. Il va afficher une autre fenêtre sur laquelle vous pouvez saisir un nom de fichier existant pour le chiffrer dans un autre fichier binaire.



Les codes complètes sont dans le fichier [TP3.py](#).

TP4 - Matplotlib

4.1 Introduction simple

Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy.

Plusieurs points intéressants de Matplotlib:

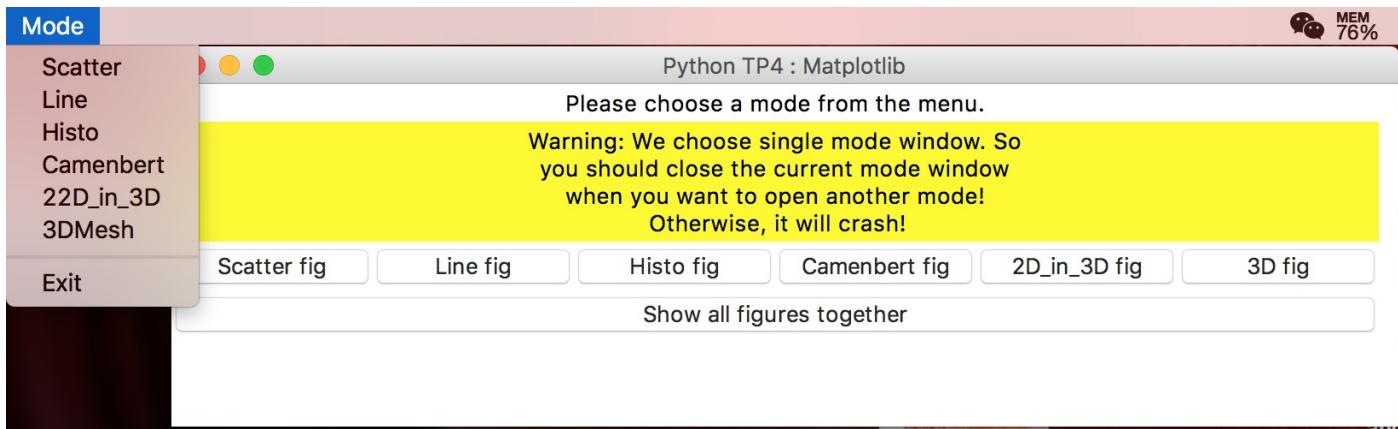
- Export possible en de nombreux formats matriciels (PNG, JPEG...) et vectoriels (PDF, SVG...)
- Documentation en ligne en quantité, nombreux exemples disponibles sur internet
- Forte communauté très active
- Interface pylab : reproduit fidèlement la syntaxe MATLAB
- Bibliothèque haut niveau : idéale pour le calcul interactif

Il peut réaliser plusieurs types de graphiques:

- Graphique linéaire
- Nuage de points
- Carte de contour
- Diagramme à barres
- L'histogramme
- Graphiques 3D,
- Même des animations graphiques et ainsi de suite.

4.2 Travail du TP

J'ai créé une interface avec des buttons et un menu en haut qui contient tous ce qu'on a fait de ce TP. L'utilisateur peut faire des opérations depuis cette interface (comme ci-dessous).



Afficher la courbe de ces données dans une fenêtre matplotlib et afficher plusieurs courbes avec styles et couleurs variés (voir fig1 et fig2), et puis modifier les noms des axes, la légende, ajouter des flèches pour montrer des zones(voir fig3):

```
def drawScatter():

    global plot
    plot.figure('Scatter fig')

    ax = plot.gca()
    ax.set_xlabel('x')
    ax.set_ylabel('y')

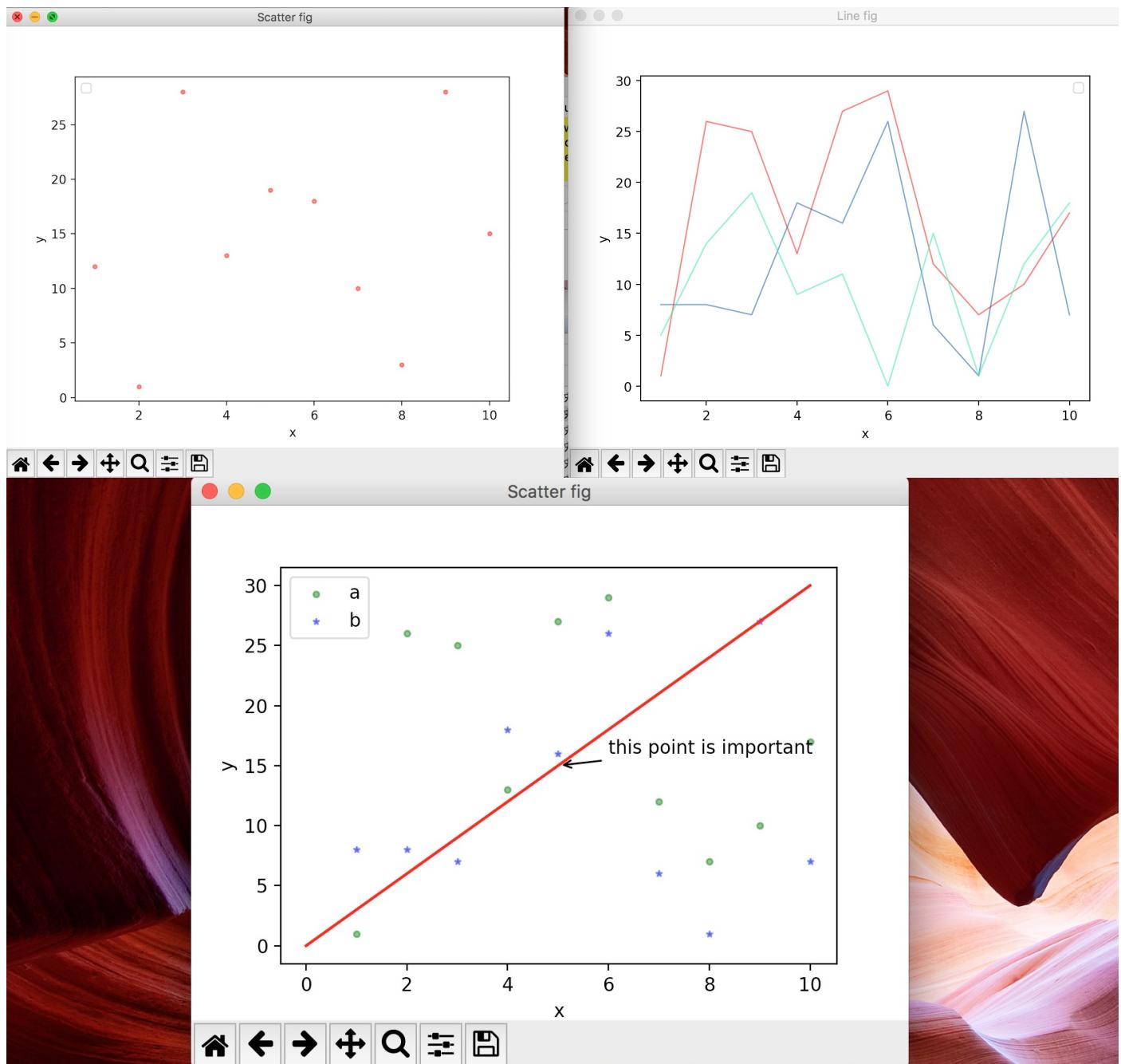
    l1 = ax.scatter(numbers_x, numbers_y, c='g', marker = 'o', s=10, alpha=0.5)
    l2 = ax.scatter(numbers_x, numbers_y2, c='blue', marker = '*', s=10, alpha=0.5)

    plot.plot([0, 10], [0, 30], color = 'red', linestyle = 'solid')
    plot.annotate(text="this point is important", xy=(5, 15), xytext=(6,
16), arrowprops={"arrowstyle":"->"})

    plot.legend(handles = [l1, l2], labels = ['a', 'b'], loc = 'best')
    plot.show()
```

```
def drawLine():
    global plot
    plot.figure('Line fig')
    # plot.xlabel('X axis')
    # plot.ylabel('Y axis')
    ax = plot.gca()
    ax.set_xlabel('x')
    ax.set_ylabel('y')

    g1 = ax.plot(numbers_x, numbers_y, c='r', linewidth=1, alpha=0.6)
    g2 = ax.plot(numbers_x, numbers_y2, color="#054E9F", linewidth=1, alpha=0.6)
    g3 = ax.plot(numbers_x, numbers_y3, color="#1DF09B", linewidth=1, alpha=0.6)
    plot.legend(handles = [g1,g2,g3], labels = ['line1', 'line2','line3'], loc =
'best')
    plot.show()
```



Afficher un histogramme et un camembert:

```
def drawHisto():

    global plot
    plot.figure('Historique Bar fig')
    ax = plot.gca()
    ax.set_xlabel('value')
    ax.set_ylabel('count')

    xticks = np.arange(1, len(numbers_x)+1)
    bar_width=0.5

    ax.bar(xticks, numbers_y, width=bar_width, edgecolor='none')
    ax.set_xticks(xticks)

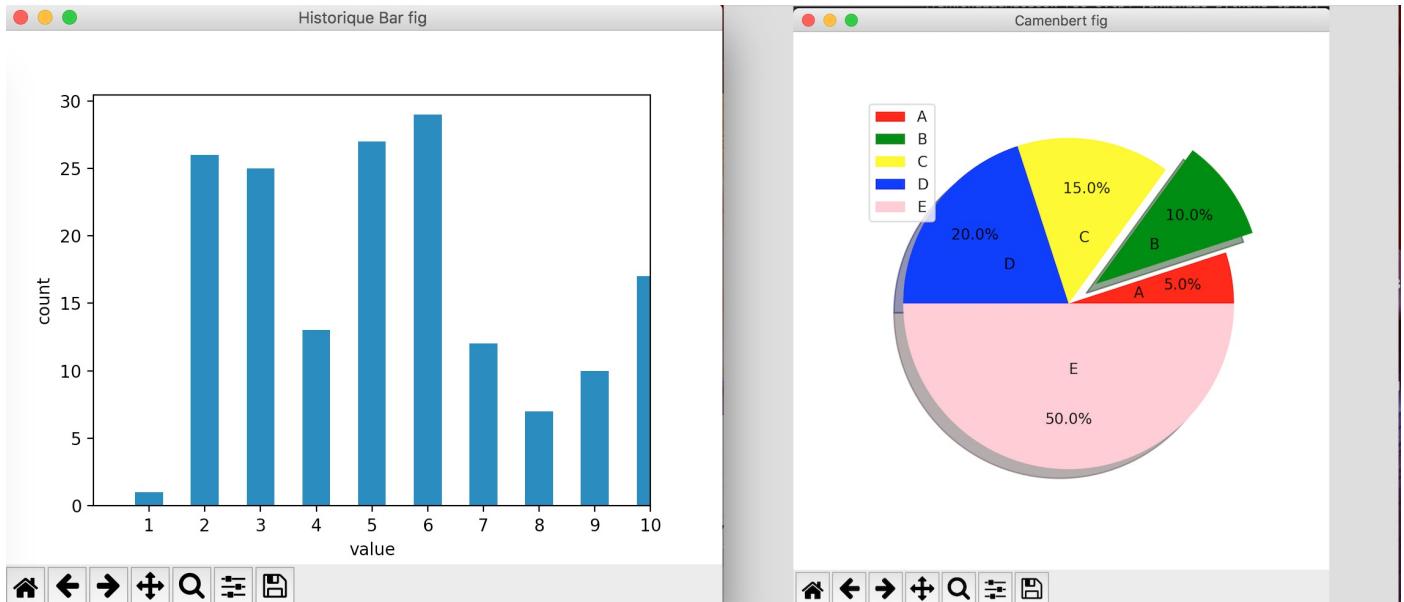
    ax.set_xticklabels(numbers_x)
    ax.set_xlim(0,len(xticks))
    plot.show()
```

```

def drawCamenbert():

    plot.figure('Camenbert fig', figsize = (5, 5))
    x = [1, 2, 3, 4, 10]
    plot.pie(x, labels = ['A', 'B', 'C', 'D', 'E'],
              colors = ['red', 'green', 'yellow', 'blue', 'pink'],
              explode = [0, 0.2, 0, 0, 0],
              autopct = lambda x: str(round(x, 2)) + '%',
              pctdistance = 0.7, labeldistance = 0.4,
              shadow = True)
    plot.legend(loc='upper left')
    plot.show()

```



Mais on a trouvé qu'on peut ouvrir qu'une image chaque fois donc on a apris comment mettre plusieurs figures dans une fenêtre en utilisant `subplot` :

```

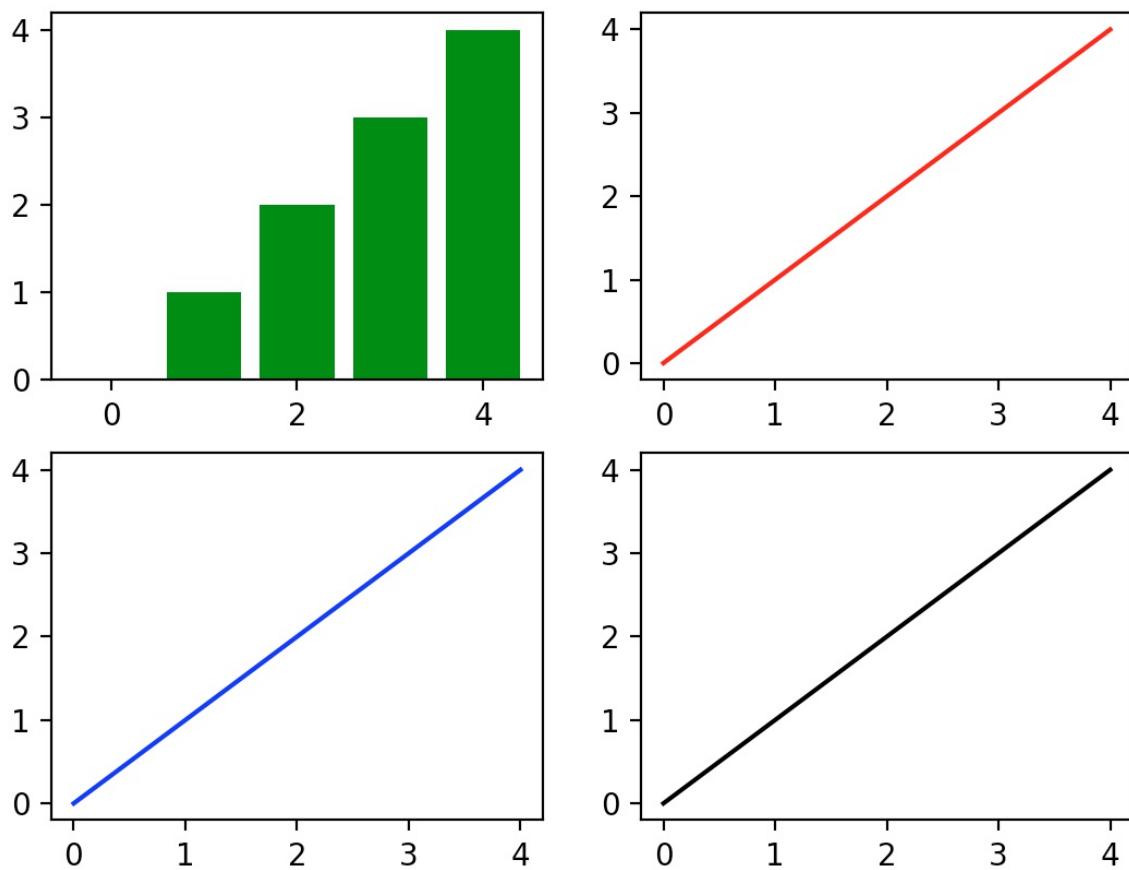
def showAllFigs():
    fig = plot.figure()
    ax1 = fig.add_subplot(223)
    ax2 = fig.add_subplot(221)
    ax3 = fig.add_subplot(222)
    ax4 = fig.add_subplot(224)

    ax1.plot(range(5), color = 'blue')
    ax2.bar(range(5), range(5), color = 'green')
    ax3.plot(range(5), color = 'red')
    ax4.plot(range(5), color = 'black')

```



Figure 1



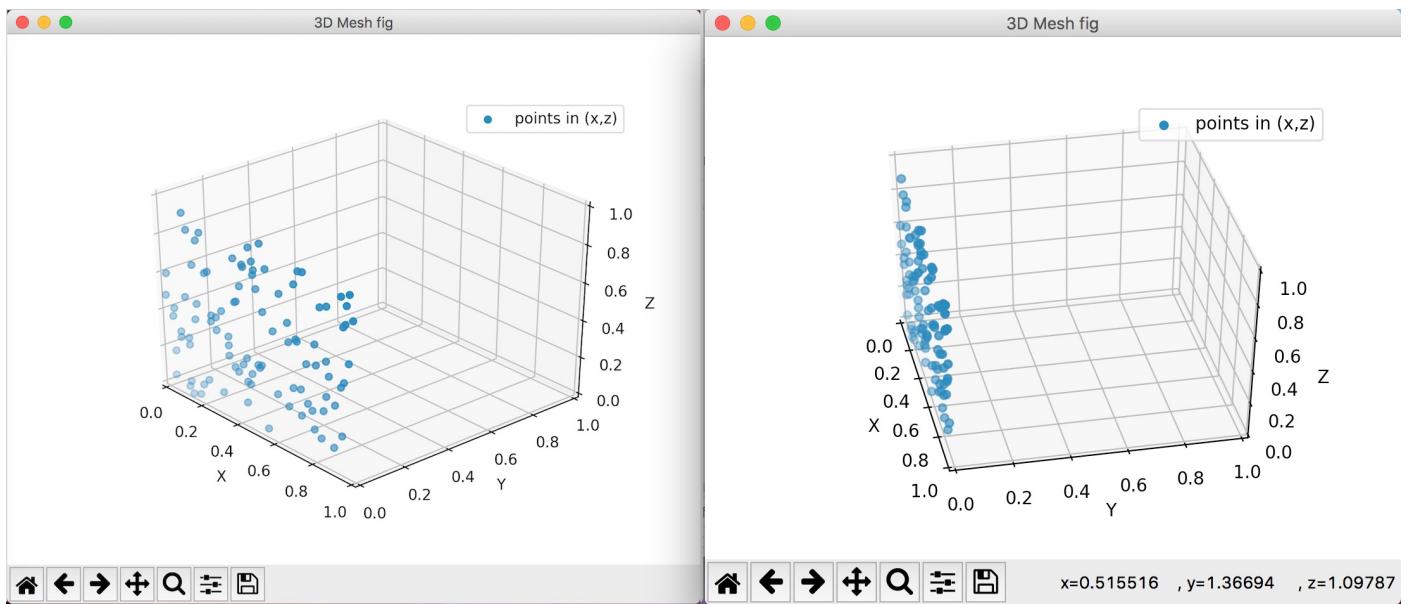
Afficher une surface 2D dans un espace 3D (mesh):

```
def mesh_2d_3d():
    fig = plt.figure('3D Mesh fig') #titre
    ax = fig.add_subplot(111, projection='3d') #une image en 3D

    x = np.random.sample(100)
    y = np.random.sample(100)

    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.set_zlim(0, 1)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    ax.scatter(x, y, zs=0, zdir='y', label='points in (x,z)')
    ax.legend()
    plt.show()
```

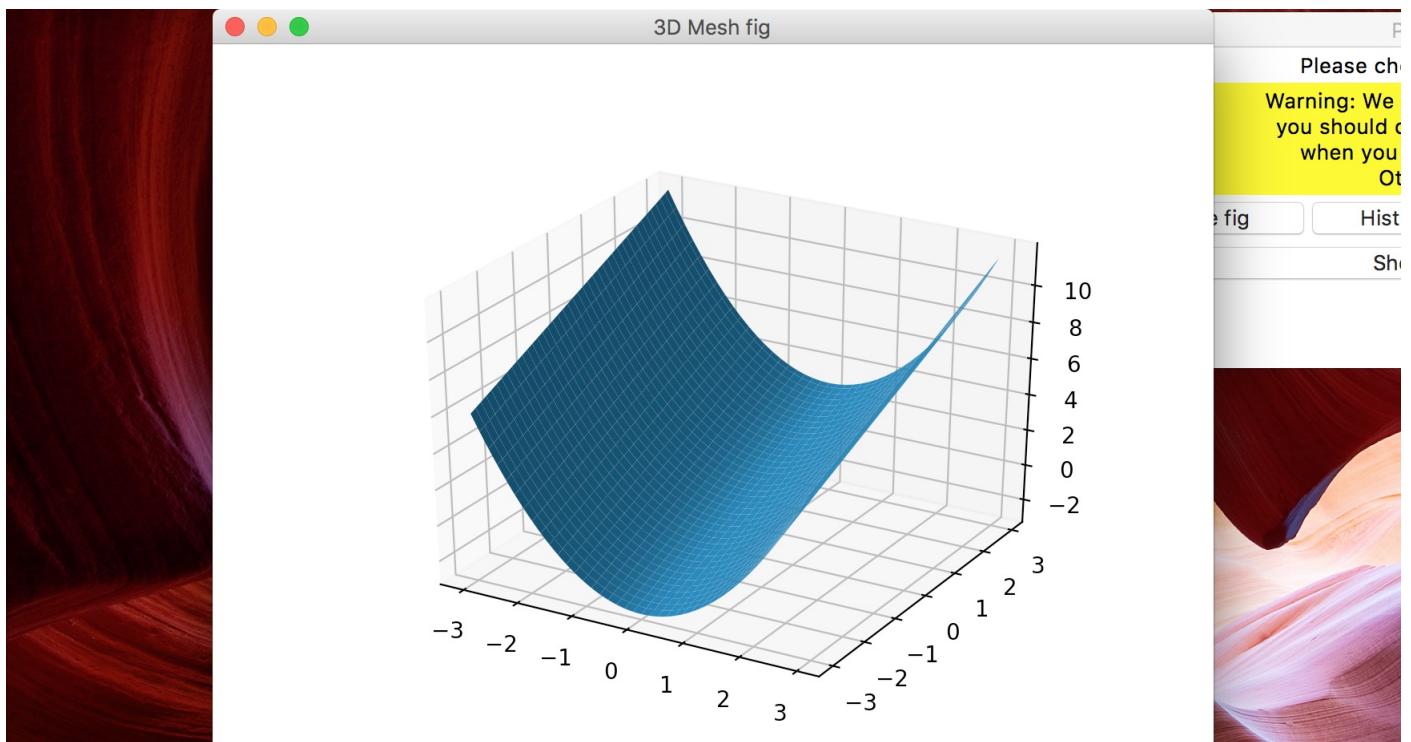


Les deux images ci-dessus sont 2D dans un espace 3D, en plus on a essayé de faire un graphe 3D dans un espace 3D:

```
def mesh_3d():
    fig = plt.figure('3D Mesh fig')
    ax = fig.add_subplot(111, projection='3d')

    x = y = np.arange(-3.0, 3.0, 0.05)
    X, Y = np.meshgrid(x, y)
    zs = np.array([fun(x,y) for x,y in zip(np.ravel(X), np.ravel(Y))])
    Z = zs.reshape(X.shape)

    ax.plot_surface(X,Y,Z)
    plt.show()
```



Conclusion

Après des recherches et études, on peut trouver que cet outil Matplotlib est très pratique, il peut faire nombreux types

de figures avec plusieurs attributs, comme couleur, forme, taille etc. Il peut bien montrer des données avec une façon visuelle.

TP5 - Base de données

5.1 Introduction simple - SQLite3

SQLite3 est une bibliothèque de gestion de bases de données relationnelle. Elle est déjà en général installée (windows, linux...). Commandes indispensables : create, select...from..., insert.

5.2 Travail du TP

5.2.1 Création de la base de donnée.

On a lit des fichiers `.csv` et parsé des informations dans ces fichiers. On a crée une base de donnée qui contient des classes correspondantes à chaque ficher, `Regions`, `Departements`, `Communes`.

Attention: Si la base de donnée existe déjà, on ne peut plus la générer à nouveau, si nécessaire supprimez-la du dossier TP5.

Par exemple:

```
# coding=utf-8
from sqlalchemy import *
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import csv
from sqlalchemy.orm import relationship
import xml.etree.ElementTree as ET

engine = create_engine('sqlite:///database_tp5.db:', echo=False)
Base = declarative_base()

class Departements(Base):
    __tablename__ = 'departements'
    code_departement = Column(String(50), primary_key=True)
    nom_departement = Column(String(50))
    nb_arrondi = Column(Integer)
    nb_canton = Column(Integer)
    nb_commune = Column(Integer)
    population_municipale = Column(Integer)
    population_totale = Column(Integer)

    code_region = Column(Integer, ForeignKey('regions.code_region'))
    region = relationship("Regions", back_populates="departements")

    communes = relationship("Communes", back_populates="departements")

    ...
Base.metadata.create_all(engine)
```

```
with open('data_Base_de_donnees/departements.csv', newline='', encoding = 'ISO-8859-1') as csvfile:
    count = 0;
    spamreader = csv.reader(csvfile, delimiter=';')
    for row in spamreader:
        count += 1;
        if count > 8:
```

```

departement = Departements()
departement.code_region = row[0]
departement.code_departement = row[2]
departement.nom_departement = row[3]
departement.nb_arrondi = row[4].replace(' ', '')
departement.nb_canton = row[5].replace(' ', '')
departement.nb_commune = row[6].replace(' ', '')
departement.population_municipale = row[7].replace(' ', '')
departement.population_totale = row[8].replace(' ', '')
session.add(departement)
session.commit()

```

The screenshot shows the DB Browser for SQLite interface with the following details:

- Toolbar:** 新建数据库(N), 打开数据库(O), 写入更改(W), 倒退更改(R).
- Database Structure Tab:**
 - Buttons: 创建表(C), 创建索引(I), 修改表, 删除表.
 - Table List:
 - communes** (CREATE TABLE communes (code_commun...))
 - departements** (CREATE TABLE departements (code_departement ...))
 - regions** (CREATE TABLE regions (code_region ...))
 - Counters: 表(3), 索引(0), 视图(0), 触发器(0).
- SQL Editor Tab:** 模式: 文本, 导入, 导出, 设为空.
- Chart View:**
 - Text: 当前在单元格中的数据的类型: 空, 0字节, 应用.
 - Chart: A scatter plot with X and Y axes ranging from 0 to 5. The data points are (0, 0), (1, 1), (2, 2), (3, 3), and (4, 4). The chart has a light gray background with a grid.
 - Controls: 列, 线形: 折线, 点形: 实心点, 图表(P), SQL 日志(L), 数据库架构(M), 远程(R).

The screenshot shows the MySQL Workbench interface. On the left, the 'Database Structure' tab is active, displaying the 'communes' table with columns: code_commune, code_departement, nom_commune, population_part, population_municipal, and population_totale. The table contains 46 rows of data. On the right, the 'Edit Database Cell' tab is active, showing a grid for editing a specific cell. The cell being edited is currently empty. Below the grid, it says '当前在单元格中的数据的类型: 空' (The type of data in the current cell is empty) and '0字节' (0 bytes). At the bottom, there are tabs for 'SQL 日志(L)', '图表(P)' (selected), '数据库架构(M)', and '远程(R)'. The status bar at the bottom right shows 'UTF-8'.

On a appris comment créer la base de données avec des classes et comment définir des attributs et des relations tels que la clé étrangère.

5.2.2 Calculer et afficher les populations totales de chaque département et région.

On fait des requêtes de la base de données en utilisant `session.query()`.

```
def comparerPopulations():
    for row in session.query(Regions).all():
        totale_region = 0
        for departement_item in row.departements:
            totale_region += departement_item.population_totale
        print(row.nom_region, totale_region)
```

```
YanwenlideMBP-3:tp5_yan yanwenli$ python3 tp5.py
[Guadeloupe 409055
Martinique 391837
Guyane 246507
La Réunion 844741
Île-de-France 12116367
Champagne-Ardenne 1375674
Picardie 1974614
Haute-Normandie 1892928
Centre 2641391
Basse-Normandie 1523247
Bourgogne 1693615
Nord-Pas-de-Calais 4127229
Lorraine 2400402
Alsace 1903801
Franche-Comté 1213499
Pays de la Loire 3765802
Bretagne 3361496
Poitou-Charentes 1844972
Aquitaine 3406433
Midi-Pyrénées 3038568
Limousin 759577
Rhône-Alpes 6557824
Auvergne 1398946
Languedoc-Roussillon 2789059
Provence-Alpes-Côte d'Azur 5039311
Corse 325510
```

5.2.3 Requêtes

Déterminer les communes ayant le même nom et un département différent. Afficher le nom de la commune suivi de la liste des n° de départements.

```
def trouverMemeNom():
    lstCommunes = session.query(distinct(Communes.nom_commune)).all() #return colonne nom_commune

    for row in lstCommunes:
        print("-----")
        # print(row[0]) #print le nom de commune
        lstDepartements = session.query(Communes.nom_commune.label('nom_commune'),
                                         Communes.code_departement.label('code_departement')).filter(Communes.nom_commune == row[0]).all()

        for d in lstDepartements:
            print(d[0],"\t\t", d[1])
```

| | |
|----------------------------|----|
| Saint-Nizier-le-Désert | 1 |
| Saint-Paul-de-Varax | 1 |
| Saint-Rambert-en-Bugey | 1 |
| Saint-Rémy | 1 |
| Saint-Rémy | 12 |
| Saint-Rémy | 14 |
| Saint-Rémy | 19 |
| Saint-Rémy | 21 |
| Saint-Rémy | 24 |
| Saint-Rémy | 71 |
| Saint-Rémy | 79 |
| Saint-Sorlin-en-Bugey | 1 |
| Saint-Sulpice | 1 |
| Saint-Sulpice | 46 |
| Saint-Sulpice | 49 |
| Saint-Sulpice | 53 |
| Saint-Sulpice | 58 |
| Saint-Sulpice | 60 |
| Saint-Sulpice | 63 |
| Saint-Sulpice | 70 |
| Saint-Sulpice | 73 |
| Saint-Trivier-de-Courtes | 1 |
| Saint-Trivier-sur-Moignans | 1 |
| Saint-Vulbas | 1 |

5.2.4 Sauvegarde et lecture de base de données (xml)

Ecrire une fonction pour sauvegarder la base dans un fichier XML et une autre pour charger la base à partir de ce fichier.

Sauvegarde au xml: une fonction pour sauvegarder la base dans un fichier XML.

```
def xml_to_database():
    tree = ET.parse('Regions.xml')
    root = tree.getroot()
    # print('root-tag:',root.tag,',root-attrib:',root.attrib,',root-text:',root.text)
    for child in root:
        region = Regions()
        region.code_region = child.find('code_region').text
        region.nom_region = child.find('nom_region').text
        region.nb_arrondi = child.find('nb_arrondi').text
        region.nb_canton = child.find('nb_canton').text
        region.nb_commune = child.find('nb_commune').text
        region.population_municipale = child.find('population_municipale').text
        region.population_totale = child.find('population_totale').text
        session.add(region)
    session.commit()
```

```

1  <?xml version="1.0" ?>
2  <mydata>
3      <row>
4          <code_region>1</code_region>
5          <nom_region>Guadeloupe</nom_region>
6          <nb_arrondi>2</nb_arrondi>
7          <nb_canton>21</nb_canton>
8          <nb_commune>32</nb_commune>
9          <population_municipale>402119</population_municipale>
10         <population_totale>409055</population_totale>
11     </row>
12     <row>
13         <code_region>2</code_region>
14         <nom_region>Martinique</nom_region>
15         <nb_arrondi>4</nb_arrondi>
16         <nb_canton></nb_canton>
17         <nb_commune>34</nb_commune>
18         <population_municipale>385551</population_municipale>
19         <population_totale>391837</population_totale>
20     </row>
21     <row>

```

Lecture depuis xml: une fonction pour charger la base à partir de ce fichier.

```

with open(nomFichier, "a") as outfile:

rows_regions = session.query(Regions).all()
outfile.write('<?xml version="1.0" ?>\n')
outfile.write('<mydata>\n')
for row in rows_regions:
    outfile.write('    <row>\n')
    outfile.write('        <code_region>%s</code_region>\n' % row.code_region)
    outfile.write('        <nom_region>%s</nom_region>\n' % row.nom_region)
    outfile.write('        <nb_arrondi>%s</nb_arrondi>\n' % row.nb_arrondi)
    outfile.write('        <nb_canton>%s</nb_canton>\n' % row.nb_canton)
    outfile.write('        <nb_commune>%s</nb_commune>\n' % row.nb_commune)
    outfile.write('        <population_municipale>%s</population_municipale>\n' %
row.population_municipale)
    outfile.write('        <population_totale>%s</population_totale>\n' %
row.population_totale)

    outfile.write('    </row>\n')
outfile.write('</mydata>\n')

```

```

b_commune, population_municipale, population_totale) VALUES (?, ?, ?, ?, ?, ?, ?)' ] [parameters: ((1', 'Guadeloupe', '2', '21', '32', '402119', '409055'), ('2', 'Martinique', '4',
None, '34', '385551', '391837'), ('3', 'Guyane', '2', None, '22', '244118', '246507'), ('4', 'La Réunion', '4', '25', '24', '835183', '844741'), ('11', 'Île-de-France', '28', '15
5', '1280', '11959887', '12116367'), ('21', 'Champagne-Ardenne', '15', '76', '1953', '1339808', '1375674'), ('22', 'Picardie', '13', '65', '2298', '1927142', '1974614'), ('23', 'H
auts-de-France', '5', '58', '1420', '1849652', '1892928') ... displaying 10 of 26 total bound parameter sets ... ('93', 'Provence-Alpes-Côte d'Azur', '18', '126', '958', '495367
5', '5039311'), ('94', 'Corse', '5', '26', '360', '320208', '325510'))] (Background on this error at: http://sqlalche.me/e/gkpj)
YanwenlideMBP-3:tp5_yan yanwenli$ 

```

5.2.5 Ajout d'une table NouvellesRegions

```

def addNouvellesRegions():
    spamreader = csv.reader(csvfile, delimiter=';')
    for row in spamreader:
        count += 1;
        if count > 8:
            print(row)
            region = Regions()
            region.code_region = row[0]
            region.nom_region = row[1]
            region.nb_arrondi = row[2].replace(' ', '')
            region.nb_canton = row[3].replace(' ', '')
            region.nb_commune = row[4].replace(' ', '')

```

```

        region.population_municipale = row[5].replace(' ', '')
        region.population_totale = row[6].replace(' ', '')
        session.add(region)
    session.commit()

```

| 名称 | 类型 | 架构 |
|-----------------------|-----------------|---|
| ▼ 表 (4) | | |
| ► 表 communes | | CREATE TABLE communes (code_commune INTEGER NOT NULL, code_ |
| ► 表 departements | | CREATE TABLE departements (code_departement VARCHAR(50) NOT N |
| ▼ 表 nouvellesRegions | | CREATE TABLE "nouvellesRegions" (code_region INTEGER NOT NULL, |
| code_region | INTEGER | `code_region` INTEGER NOT NULL |
| nom_region | VARCHAR (...) | `nom_region` VARCHAR (50) |
| nb_commune | INTEGER | `nb_commune` INTEGER |
| population_municipale | INTEGER | `population_municipale` INTEGER |
| population_totale | INTEGER | `population_totale` INTEGER |
| ► 表 regions | | CREATE TABLE regions (code_region INTEGER NOT NULL, nom_region |
| 索引 (0) | | |

TP6 - Numpy et Scipy

Dans ce TP, on a bien compris des fonctionnalités et des caractéristiques de Numpy et Scipy en réalisant tous les exercices.

Nous avons créé trois fichiers: [tp6.py](#) (Numpy), [tp6_scipy.py](#) (Scipy) [tp6_matplotlib_image.py](#) (qui utilise PIL pour réaliser des opérations sur l'image)

6.1 Numpy

6.1.1 Introduction simple

NumPy est le paquet fondamental du calcul scientifique et une bibliothèque tierce essentielle à l'analyse de données en Python.

L'émergence de np résout dans une certaine mesure le problème des performances informatiques médiocres en Python, en même temps le fournit des types de données plus précis. On peut dire que NumPy est la bibliothèque la plus élémentaire de traitement de données ou de calcul scientifique.

Il contient:

- Un puissant objet tableau N-dimensionnel
- Fonctions sophistiquées (diffusion)
- Outils d'intégration de code C / C ++ et Fortran
- Algèbre linéaire utile, transformée de Fourier et capacités de nombres aléatoires

6.1.2 Travail du TP

Afficher des attributs du tableau

```

def creationA():
    #Créer un tableau de dimension 3 avec un shape de (4, 3, 2) rempli avec des
    #nombres aléatoires.
    a1 = np.random.empty(0,10,size=[4,3,2])

    #Vous afficherez les attributs du tableau : ndim, shape, size, dtype, itemsize,
    #data.
    print(a1)
    print("ArrayType: ",type(a1))
    print("DataType: ",a1.dtype)

```

```

print("ArraySize: ",a1.size)
print("ItemSize: ",a1.itemsize)
print("ArrayShape: ",a1.shape)
print("data: ",a1.data)
print("ArrayDimension: ",a1.ndim)

```

Résultats: int:

```

[[[1 0]
 [6 1]
 [1 7]]

 [[4 3]
 [8 7]
 [6 2]]

 [[3 2]
 [4 1]
 [7 8]]

 [[5 7]
 [5 8]
 [3 4]]]

ArrayType: <class 'numpy.ndarray'>
DataType: int64
ArraySize: 24
ItemSize: 8
ArrayShape: (4, 3, 2)
data: <memory at 0x1040f15e8>
ArrayDimension: 3
YanwenlideMBP-3:tp6_yan yanwenli$ █

```

float:

```

[[[0.0000000e+000 0.0000000e+000 2.15871368e-314 2.21554614e-314]
 [2.21569853e-314 2.21867788e-314 2.21866905e-314 2.21866800e-314]
 [2.21866908e-314 2.21569848e-314 2.21866911e-314 2.21569859e-314]]

[[2.21866914e-314 2.21865932e-314 2.21866917e-314 2.21866920e-314]
 [2.21866924e-314 2.21568230e-314 2.21569850e-314 2.21866927e-314]
 [2.21867791e-314 2.21867794e-314 2.21779979e-314 1.66880539e-308]]]

ArrayType: <class 'numpy.ndarray'>
DataType: float64
ArraySize: 24
ItemSize: 8
ArrayShape: (2, 3, 4)
data: <memory at 0x10b87bd68>
ArrayDimension: 3

```

Des opérations sur array avec Numpy

On a créé deux matrices pour faire le `+`, `dot`, `*`, `multiply` et `transposer`.

```

b1_arr = np.arange(0,9).reshape(3,3)    # ne contient pas 9 (0-8)
b2_arr = np.arange(2,11).reshape(3,3)   # ne contient pas 11 (2-10)
b1_mat = np.mat(b1_arr)
b2_mat = np.mat(b2_arr)

```

Pour bien comprendre les différences des opérations pour Array et Matrice, on a testé avec les deux:

```

print("----- array -----")
print("b1_mat\n",b1_arr)
print("b2_mat\n",b2_arr)

print("+\n", b1_arr + b2_arr)

```

```
print("dot\n", np.dot(b1_arr, b2_arr))
print("*\n", (b1_arr) * (b2_arr))
print("multiply\n", np.multiply(b1_arr, b2_arr))
print("transposer\n", np.transpose(b1_arr))
```

```
----- array -----
b1_mat
[[0 1 2]
 [3 4 5]
 [6 7 8]]
b2_mat
[[ 2  3   4]
 [ 5  6   7]
 [ 8  9  10]]
+
[[ 2  4   6]
 [ 8 10  12]
 [14 16  18]]
dot
[[ 21  24  27]
 [ 66  78  90]
 [111 132 153]]
*
[[ 0  3   8]
 [15 24  35]
 [48 63  80]]
multiply
[[ 0  3   8]
 [15 24  35]
 [48 63  80]]
transposer
[[0 3 6]
 [1 4 7]
 [2 5 8]]
```

```
----- matrice -----
print("b1_mat\n", b1_mat)
print("b2_mat\n", b2_mat)
print("+\n", b1_mat + b2_mat)
print("dot\n", np.dot(b1_mat, b2_mat))
print("*\n", (b1_mat) * (b2_mat))
print("multiply\n", np.multiply(b1_mat, b2_mat))
print("transposer of b1_mat: \n", np.transpose(b1_mat))
```

```

----- matrice -----
b1_mat
[[0 1 2]
 [3 4 5]
 [6 7 8]]
b2_mat
[[ 2  3   4]
 [ 5  6   7]
 [ 8  9  10]]
+
[[ 2  4   6]
 [ 8 10  12]
 [14 16  18]]
dot
[[ 21  24  27]
 [ 66  78  90]
 [111 132 153]]
*
[[ 21  24  27]
 [ 66  78  90]
 [111 132 153]]
multiply
[[ 0  3   8]
 [15 24  35]
 [48 63  80]]
transposer of b1_mat:
[[0 3 6]
 [1 4 7]
 [2 5 8]]

```

Selon les résultats, on peut faire la conclusion:

- **dot** : Pour un tableau de rang 1, effectuez la multiplication de position correspondante puis faire la somme; pour un tableau à deux dimensions dont le rang n'est pas égal à 1, effectuez la multiplication matricielle.
- ***** : Effectuer la multiplication de position correspondante sur le tableau; effectuer une multiplication matricielle sur des matrices.

Autres opérations sur matrice

Calculer le déterminant, l'inverse, les valeurs et vecteurs propres d'une matrice.

```

print("det of b1_mat: \n", np.linalg.det(b1_mat))
#le déterminant égale à 0, ce matrice est invertible
try:
    inverse = np.linalg.inv(b2_mat)
    print("inverse of b2_mat: \n", inverse)
except np.linalg.LinAlgError:
    print("Not invertible. Skip this one.") #Si non invertible. Passer à la suite
    pass
else:
    print("ok")
eig1,eig2 = np.linalg.eig(b1_mat)
print("the eig of mat is:\n",eig1)
print("the feature vector of mat is:\n",eig2)

```

```

det of b1_mat:
0.0
Not invertible. Skip this one.
the eig of mat is:
[ 1.33484692e+01 -1.34846923e+00 -2.48477279e-16]
the feature vector of mat is:
[[ 0.16476382  0.79969966  0.40824829]
 [ 0.50577448  0.10420579 -0.81649658]
 [ 0.84678513 -0.59128809  0.40824829]]

```

6.2 Scipy

6.2.1 Introduction simple

SciPy est un module dans Python construit sur Numpy qui intègre plusieurs algorithmes mathématiques et des fonctions pratiques. En fournissant aux utilisateurs des commandes et des classes de haut niveau, SciPy augmente considérablement la possibilité de manipuler et de visualiser les données dans des sessions interactives Python.

SciPy contient des modules:

- Optimisation
- Algèbre linéaire
- Intégration
- Interpolation
- Fonctions spéciales
- Transformée de Fourier rapide
- Traitement du signal et de traitement d'images
- Résolution d'équations différentielles ordinaires
- D'autres calculs couramment utilisés en science et en ingénierie.

6.2.2 Travail du TP

Approcher un ensemble de points par une courbe

Pour une solution polynomiale simple, on peut utiliser simplement la fonction numpy: `polyfit (x, y, degree)`. Pour le fitting exponentiel et puissant, on peut utiliser `curve_fit` dans `scipy.optimize`

Donc on a fait des exercices avec deux façons:

```

def polyfitting_1():
    x = np.arange(1, 16, 1)
    num = [4.00, 5.20, 5.900, 6.80, 7.34,
           8.57, 9.86, 10.12, 12.56, 14.32,
           15.42, 16.50, 18.92, 19.58, 20.00]
    y = np.array(num)

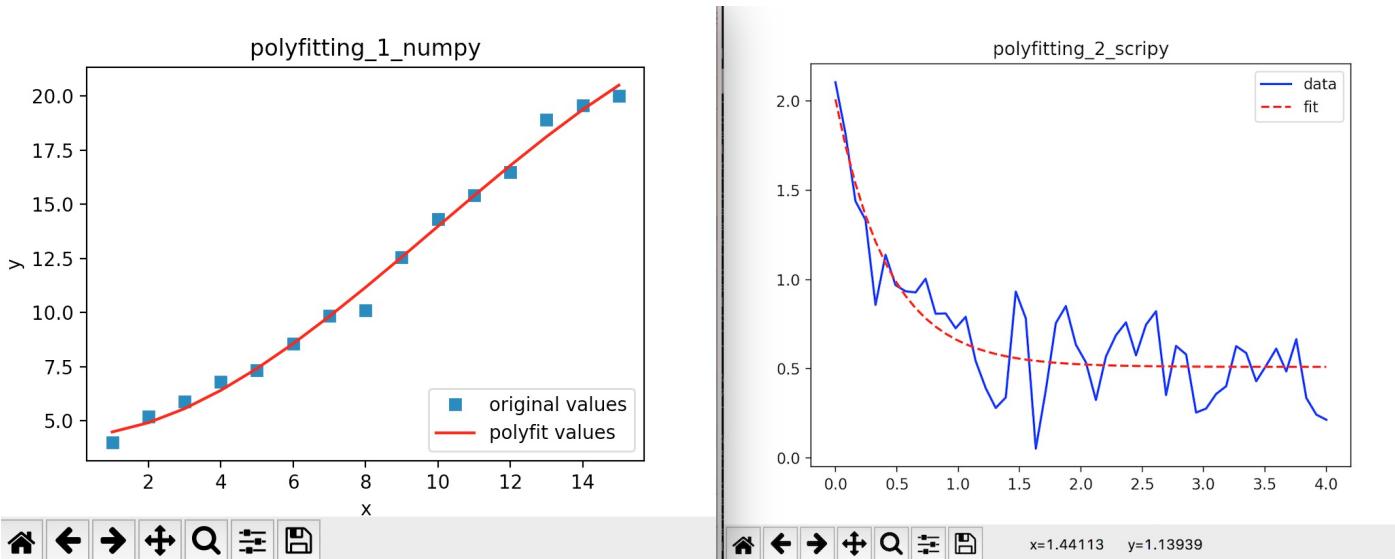
    #Fit avec un polynôme cubique
    f1 = np.polyfit(x, y, 3)
    p1 = np.poly1d(f1)
    print(p1)

    #Fit y value
    # methode2: yvals=np.polyval(f1, x)
    yvals = p1(x)

```

```
#Dessiner un graphe
plot1 = plt.plot(x, y, 's', label='original values')
plot2 = plt.plot(x, yvals, 'r', label='polyfit values')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc=4) #legend:right-down
plt.title('polyfitting_1_numpy')
plt.show()
plt.savefig('test.png')
```

```
def polyfitting_2():
    xdata = np.linspace(0, 4, 50)
    y = func(xdata, 1.5, 2.3, 0.5)
    ydata = y + 0.2 * np.random.normal(size=len(xdata))
    plt.plot(xdata,ydata,'b-',label='data')
    #Ajustement des moindres carrés non linéaire
    popt, pcov = curve_fit(func, xdata, ydata)
    #popt[], trois inconnus en attente de résolution a,b,c
    y2 = [func(i, popt[0],popt[1],popt[2]) for i in xdata]
    plt.plot(xdata,y2,'r--',label='fit')
    plt.title('polyfitting_2_scipy')
    plt.legend()
    plt.show()
    print(popt)
```



6.3 Traitement sur images

Pour mieux comparer, on a récupéré tous les résultats ensemble: Pour lire une image jpeg: `matplotlib.imread` ne marche que sur le format `.png` mais pas sur `.jpg`, alors après des tests sur toutes les façons possibles, on a choisi `PIL`.

et afficher l'image originale et réduite en taille

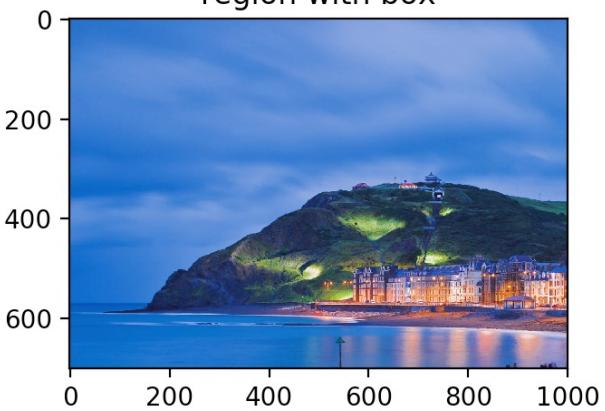
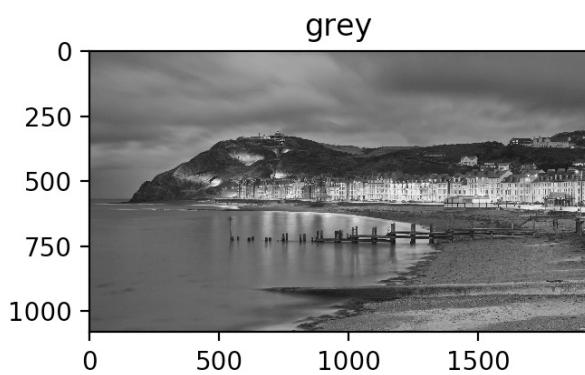
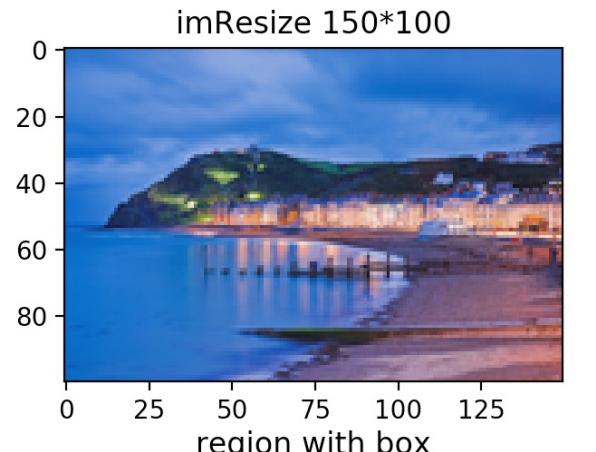
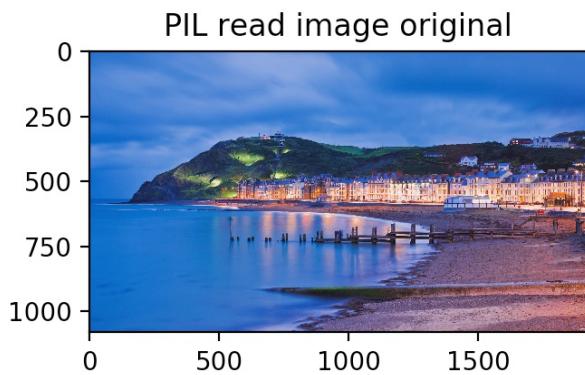
```
test_image = Image.open(imagepath_2)
imResize = test_image.resize((150,100), Image.ANTIALIAS)
l_image = test_image.convert('L')
box = (0, 0, 1000, 700)
region = test_image.crop(box)

plt.subplot(221)
plt.title('PIL read image original')
plt.imshow(test_image)
```

```

plt.subplot(222)
plt.title('imResize 150*100')
plt.imshow(imResize)
plt.subplot(223)
plt.title('grey')
plt.imshow(l_image)
plt.subplot(224)
plt.title('region with box')
plt.imshow(region)
plt.show()

```



TP7 - Serveur et page web (Django)

Nous avons essayé de lancer un server avec les codes fournis.

The screenshot shows a browser window with the URL `localhost:8888/index.py`. The page contains a form with a text input field labeled "Votre nom" and a button "Envoyer information au serveur". Below the form, there is some server log output:

```

→ Python_TP python3 TP7.py
Serveur actif sur le port : 8888
127.0.0.1 -- [23/Nov/2018 22:21:51] "POST /index.py HTTP/1.1" 200 -
127.0.0.1 -- [23/Nov/2018 22:21:55] "GET /index.py HTTP/1.1" 200 -
127.0.0.1 -- [23/Nov/2018 22:21:57] "POST /index.py HTTP/1.1" 200 -

```

Pour la partie login et afficher les données, nous avons utilisé `Django`.

Login et afficher les données

Afin de lancer le projet django, il faut entrer dans le dossier **TP7** et exécuter la commande `python3 manage.py runserver` (ou `manage.py runserver` sous Windows).

Nous avons fait une interface web pour que les utilisateurs puissent se connecter au système avec login et mot de passe. Le URL de la page login est <http://localhost:8000/site>.

← → ⌂ ⓘ 127.0.0.1:8000/site/

Login:

Username:

Password:

- **Username** : admin
- **Password** : password

Si le login ou le mot de passe n'est pas correct, il y aura une message d'erreur.

Login:

- Username or password error

Username:

Password:

Après accéder à la page d'accueil (<http://localhost:8000/site/index>) avec le login et mot de passe, il y aura une liste des informations des étudiants. Nous avons stocké ces informations dans la base de données et les récupérer avec django.

Le statut de login est stocké dans les COOKIES donc vous pouvez rester connecté pendant 3600 secondes.

Nous avons créé des `template` pour afficher les résultats et des modèles pour stocker ou récupérer les données depuis la base de données.

```
def index(req):
    username = req.COOKIES.get('username', '')
    etudiants = Etudiant.objects.all()

    return render_to_response('index.html', {'username':username,
    'etudiants':etudiants})
```

```
<table>
  <thead>
    <td>Nom</td>
    <td>Prenom</td>
    <td>Age</td>
  </thead>
  {% for e in etudiants %}
```

```

<tr>
    <td>{{e.nom}}</td>
    <td>{{e.prenom}}</td>
    <td>{{e.age}}</td>
</tr>
{% endfor %}
</table>

```

Les codes complètes sont dans le dossier [TP7/tp7_site](#).

TP8 - Prog. asynchrone et fourmis

Prog. asynchrone

Nous avons testé une même calcul ci-dessous avec `multithread` et `multiprocessing`.

```

def calcul_long():
    n = 1E7
    while n>0:
        n -= 1

```

Multi-thread (5 threads)

```
[→ TP8 git:(master) ✘ python3 tp8_threads_processus.py
Starting Thread-1
Starting Thread-2
Starting Thread-3
Starting Thread-4
Starting Thread-5
Exiting Thread-1
Exiting Thread-3
Exiting Thread-4
Exiting Thread-5
Exiting Thread-2
Exiting Main Thread
Line cpu 3.318974018096924
```

Multi-processing (5 processus)

```
[→ TP8 git:(master) ✘ python3 tp8_threads_processus.py
Parent process 22110.
Waiting for all subprocesses done...
pid:22111
pid:22112
pid:22113
pid:22114
pid:22115
All subprocesses done.
Line cpu 1.0485477447509766
```

Pour ce calcul, multi-processing est plus vite que multi-threads.

Peinture avec fourmis

Afin de faire dessiner, nous avons utilisé `numpy` pour générer une matrice de dimension `500*500*3` pour stocker les pixels. Et nous avons utiliser `scipy` pour stocker la matrice dans une fichier image.

Pour les paramètres, nous avons utilisé un fichier `.xml` pour passer les paramètres au programme.

La commande: `python3 ant.py nom_du_fichier.xml`

exp:

```
→ Python_TP python3 ant.py  
Il faut indiquer un fichier .xml pour les paramètres!  
→ Python_TP python3 ant.py fourmis.xml
```

Le fichier `fourmis.xml` :

```
<parametres nombre="2" iteration="100000">  
<fourmis>  
    <couleur_deposee>192, 0, 255</couleur_deposee>  
    <couleur_suivi>255, 155, 3</couleur_suivi>  
    <proba>0.04,0.95,0.01</proba>  
    <type>0</type>  <!-- 0 => Do, 1 =>Dd  -->  
    <proba_suivi>0.78</proba_suivi>  
</fourmis>  
<fourmis>  
    <couleur_deposee>255, 155, 3</couleur_deposee>  
    <couleur_suivi>76,68,181</couleur_suivi>  
    <proba>0.01,0.98,0.01</proba>  
    <type>1</type>  <!-- 0 => Do, 1 =>Dd  -->  
    <proba_suivi>0.90</proba_suivi>  
</fourmis>  
</parametres>
```

- nombre: le nombre de fourmis
- iteration: le nombre d'itération
- couleur_deposee: tableau de couleur RGB déposée , séparé par `,`
- couleur_suivi: tableau de couleur RGB suivie , séparé par `,`
- proba: les probabilités qui régissent le mouvement de la fourmi : (Pg, Pd, Pt)
- type: - le type de mouvement : 0 => Do, 1 => Dd
- proba_suivi: la probabilité de suivre la coureuse suivie (si elle est trouvée) : Ps

Nous n'avons pas finaliser la partie convolution, donc la taille de chaque fourmis est toujours `1`.

Après l'exécution , il va générer un fichier `ant.jpg` qui stocke les movements des fourmis.

Le code est dans le fichier `TP8/ant.py` avec les commentaires.

Une exemple:

