TP 3 INFO Modules et bibliothèques

Dans ce TP, nous allons étudier les différentes manières d'importer un module en Python, puis nous travaillerons succinctement avec quelques bibliothèques classiques.

Partie I Importer et utiliser une bibliothèque

En informatique, une bibliothèque (aussi appelée module ou librairie) est un ensemble de codes pré-écrits définissant des objets et des fonctions. Celles-ci sont regroupées et mises à disposition pour être utilisées par les programmeurs qui n'auront ainsi plus besoin de les réécrire. Les librairies permettent d'accéder à des outils supplémentaires pour traiter plus facilement des problématiques spécifiques (générateur aléatoire, gestion de données, représentation graphique, etc.).

Pour utiliser un module en Python, il faut commencer par l'importer (ou la charger) en utilisant le mot-clé `import`. Dans la suite, nous allons illustrer les différentes manières d'importer un module en donnant des exemples avec la librairie `math` qui permet, par exemple, d'utiliser les fonctions mathématiques classiques en Python.

Remarque 1 : Le programme précise que vous devez savoir importer un module avec les syntaxes présentées dans les deux sous-parties suivantes, mais que vous n'avez pas à connaître les fonctions des différents modules : leur utilisation dans un sujet devra être accompagné de la documentation utile.

I.A - Importer la totalité d'un module

La première possibilité est d'importer la totalité de la librairie avec la syntaxe ci-dessous.

```
>>> import math
```

Une fois que la librairie est chargée, vous pouvez consulter tous ses composants et leur fonctionnement en utilisant la fonction d'aide `help()`.

```
help(math)
```

Avec cette première méthode d'importation, pour utiliser une fonction d'un module, il sera nécessaire de préfixer le nom de la fonction par le nom du module dont elle est issue, afin que Python sache où la trouver.

```
>>> math.exp(1)
2.718281828459045
```

Il est possible d'utiliser un alias avec le mot clé `as` afin de raccourcir la syntaxe (ce qui est utile notamment quand le nom du module est long) lorsqu'on appelle une fonction.

```
>>> import math as m
>>> m.cos(m.pi)
-1.0
```

I.B - Importer quelques fonctions d'un module

La seconde possibilité est de se limiter à importer seulement les objets dont nous avons besoin en utilisant la syntaxe ci-dessous.

```
>>> from math import cos, sin, pi
```

Lorsqu'on utilise cette méthode d'importation, Python se souvient du module duquel proviennent les éléments importés : il n'est donc pas nécessaire de le préciser en préfixe.

```
>>> pi
3.141592653589793
>>> cos(sin(pi))
1.0
```

Partie II Simuler une expérience aléatoire : le module `random`

Dans cette partie, nous allons utiliser la librairie `random` pour simuler un expérience aléatoire.

Question 1 : Importer le module `random` avec l'alias `rd`.

Question 2 : En consultant l'aide de la librairie `random`, trouver une fonction permettant de générer aléatoirement un entier compris dans un intervalle de la forme $[a, b]$ où a et b sont des entiers.

Dans la suite de cette partie, on considère l'expérience aléatoire suivante : on effectue une succession de lancers d'un dé jusqu'à obtenir un 6.

Question 3 : Écrire un fonction `sim() -> list` qui renvoie une liste contenant les résultats successifs du dé lors de la réalisation de l'expérience aléatoire décrite ci-dessus. Voici deux exemples.

```
>>> sim()
[5, 4, 1, 3, 6]
>>> sim()
[2, 1, 3, 5, 5, 4, 1, 3, 5, 6]
```

Question 4 : En utilisant votre fonction précédente, donner une estimation du nombre de fois en moyenne où il faut lancer un dé pour obtenir un 6 pour la première fois.

Partie III Manipuler des données : le module csv

Dans cette partie, nous allons calculer quelques statistiques sur les données récupérées dans un fichier dont l'extension est csv. Pour charger ce type de fichier, nous utiliserons le module `csv`.

Pour commencer, récupérez le fichier `2017_Pop_Par_Dep.csv` dans la rubrique informatique du site se trouvant à l'adresse `http://vonbuhren.free.fr`, puis placez ce dernier au même emplacement que votre fichier Python. Le fichier `2017_Pop_Par_Dep.csv` contient des informations sur la population française en 2017 par département. Chaque ligne du fichier correspond à un département français et elle est composée de différentes informations, séparées par des virgules, sur ce département. Les informations sont les suivantes :

- `NUM` : numéro du département ;
- `NOM` : nom du département ;
- `COM` : nombre de communes du département ;
- `POP` : population du département.

Utilisez le code ci-dessous pour charger les données du fichier dans une liste.

```
import csv

with open('2017_pop_dep.csv') as fichier_csv:
    reader = csv.DictReader(fichier_csv, delimiter=',')
    lst = list(reader)
```

Chaque élément de la liste `lst` est un dictionnaire dont les clés sont `'NUM'`, `'NOM'`, `'COM'` et `'POP'`.

```
for dep in lst:
    print(dep['NUM'], dep['NOM'], dep['COM'], dep['POP'])
```

ATTENTION : Par défaut, chacune des données est enregistrée en Python dans une chaîne de caractères. Il faut donc effectuer une conversion en entier pour pouvoir faire des opérations arithmétiques sur les données numériques.

Question 5 : Déterminer le nombre de communes en France en 2017.

Question 6 : Déterminer la population totale en France en 2017.

Question 7 : En 2017, quel est le département français le plus peuplé ? le moins peuplé ?

Question 8 : Calculer le nombre moyen d'habitants par département français en 2017.

Partie IV Effectuer des représentations graphiques : le module `Matplotlib`

Dans cette partie, nous allons effectuer des représentations graphiques en utilisant la librairie `matplotlib.pyplot`.

Question 9 : Importer le module `matplotlib.pyplot` avec l'alias `plt`.

IV.A - Tracer la courbe représentative d'une fonction

Pour tracer une courbe, on utilise la fonction `plt.plot` qui prend en argument deux listes de même taille : celle des abscisses et celle des ordonnées, puis trace la ligne brisée passant par les points donnés en argument.

```
X = [0, 1, 2, 3, 4, 5] # Liste des abscisses
Y = [0, 2, 3, -1, 1, 1] # Liste des ordonnées
plt.plot(X, Y) # Trace la courbe dans la fenêtre graphique
plt.show() # Affiche la fenêtre graphique
```

Pour tracer précisément une fonction, il suffit de prendre pour X une subdivision régulière contenant un grand nombre de points.

Question 10 : Tracer la courbe représentative de la fonction exponentielle sur l'intervalle $[0,1]$.

Question 11 : Tracer sur un même graphique la courbe représentative de la fonction $f_n : x \mapsto \left(1+\frac{x}{n}\right)^n$ pour chaque entier $n \in [\![1,10]\!]$ et celle de la fonction exponentielle sur l'intervalle $[0,1]$.

IV.B - Tracer un histogramme pour représenter des données

Pour tracer un histogramme à partir d'un jeu de données, on utilise la fonction `plt.hist` qui prend en argument une liste contenant les données, puis trace un histogramme associé.

```
data = [0, 1, 0, 2, 0, 2, 3, 1, 2]
plt.hist(data) # Trace l'histogramme dans la fenêtre graphique
plt.show() # Affiche la fenêtre graphique
```

Dans certain cas, afin d'améliorer la clarté de l'histogramme, il peut être utile d'ajuster la largeur des barres en utilisant l'option `rwidth` de la fonction `plt.hist`.

```
plt.hist(data, rwidth=0.8)
```

Question 12 : En reprenant les données de la partie précédente, tracer un histogramme représentant la population par département.

Question 13 : En utilisant la commande d'aide de Python, déterminer l'utilité des fonctions suivantes :

`plt.xlabel`, `plt.ylabel`, `plt.title`.

Question 14 : Ajouter une légende et un titre à l'histogramme que vous avez généré précédemment.