PROGRAMMES, ENTRÉES ET SORTIES

1 Programmes

Un *programme* est une suite d'instructions, qui va donc réaliser des calculs lorsqu'il sera exécuté.

1. Déterminer sur papier ce que le programme suivant va afficher :

```
a = 84
b = 52
while a != b:
  if a > b:
    a = a - b
  else:
    b = b - a
print(a)
```

2. Exécuter ce programme dans Pyzo et observer que le résultat prévu est bien affiché dans la console.

2 Fonctions

Une *fonction* est un programme auquel on attribue un nom, qui prend en entrée des arguments, et qui renvoie en sortie une valeur de retour. Une fois qu'une fonction est écrite, elle peut être appelée, dans la console ou dans d'autres fonctions. Par exemple, la fonction suivante renvoie la liste obtenue en sommant termes à termes deux listes supposées de même longueur :

```
def plus(L1, L2):
  n = len(L1)
  R = []
  for i in range(n):
    x = L1[i] + L2[i]
    R.append(x)
  return R
```

1. Le site `pythontutor.com` permet de visualiser l'exécution pas à pas de programmes Python. Entrer sur ce site la définition ci-dessus de la fonction `plus`, suivie de l'appel

```
plus([0, 3, 6], [2, 8, 1])
```

   puis observer et comprendre l'exécution pas à pas du programme.

   Remarque : On pourra penser à utiliser `pythontutor.com` pour observer l'exécution pas à pas des programmes vus en cours ou en TP.

2. Que se passe-t-il quand la fonction `plus` est appelée sur des listes de longueurs différentes ?

3. Écrire (dans Pyzo) une fonction `fois` prenant en argument deux listes supposées de même longueur, et renvoyant la liste obtenue en les multipliant termes à termes.

4. Écrire une fonction `somme` prenant en argument une liste et renvoyant la somme de ses éléments.

5. Il ne faut pas confondre le mot-clé `return`, qui permet à une fonction de renvoyer une valeur, avec le mot-clé `print`, qui se contente de réaliser un affichage dans la console.

   Écrire une fonction `moyenne`, prenant en argument une liste et renvoyant la moyenne de ses termes. On utilisera dans le corps de cette fonction un appel à la fonction `somme`, et on vérifiera que cet appel ne marche pas lorsque le `return` de la fonction `somme` est remplacé par un `print`.

6. Écrire une fonction `contient` prenant deux arguments : une liste L et un élément x, et renvoyant `True` si x est dans L et `False` sinon (ce test est directement réalisé par `x in L`, qu'on s'interdira d'utiliser dans cette question).

   On testera notamment cette fonction avec l'appel `contient([0,3,5], 3)`.

7. Dans le meilleur cas, quand l'élément recherché est en tête de liste, le coût en temps de la fonction `contient` ne dépend pas de la longueur de la liste, on parle alors de **coût constant**. Dans le pire cas, quand la liste ne contient pas l'élément recherché, celle-ci est entièrement parcourue et le coût en temps est donc proportionnel à sa longueur, on parle de **coût linéaire**. Sur le même modèle, quel est le coût en temps des fonctions précédentes ?

3 Lecture de fichier

Les fonctions suivantes permettent de lire dans un fichier en Python :

- `f=open('nom_du_fichier.txt','r')` permet d'ouvrir un fichier en lecture (*read*), sous le nom de variable `f`.
- `f.readlines()` renvoie la liste des lignes présentes dans le fichier `f`, chaque ligne étant représentée sous forme de chaîne de caractères.
- `ch.strip()` renvoie la chaîne de caractère obtenue à partir de la chaîne `ch` en supprimant tous les espaces et caractères spéciaux, tels que les retours à la ligne.
- `ch.split(';')` renvoie la liste des sous-chaînes apparaissant dans la chaîne `ch` séparées par le caractère `;`
- `f.close()` ferme le fichier `f` (ce qu'il faut toujours faire une fois qu'on a fini de travailler sur un fichier qu'on a ouvert).

1. Placer les fichiers `profits.txt` et `notes.txt`, disponibles sur cahier de prepa, dans le répertoire de travail de Pyzo (dont le chemin est donné en tapant `cd` dans la console).

2. le fichier `profits.txt` contient plusieurs lignes donnant chacune la population d'une ville et le profit réalisé dans cette ville.

   Le programme suivant permet de lire ce fichier et de créer une liste `x` contenant les populations et une liste `y` contenant les profits présents dans le fichier.

```
f = open('profits.txt','r')
lignes = f.readlines()
x = []
y = []

for ligne in lignes:
    ligne = ligne.strip()
    population, profit = ligne.split(';')
    x.append(float(population))
    y.append(float(profit))
f.close()
```

Compléter ce programme pour afficher dans la console la somme des profits réalisés.

3. Les n points expérimentaux $(x_1, y_1), \dots, (x_n, y_n)$ peuvent être approchés par la droite d'équation $y = ax + b$, avec

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \quad \text{et} \quad b = \overline{y} - a\overline{x},$$

où $\overline{x}$ désigne la moyenne des $(x_i)_{1\leq i\leq n}$ et $\overline{y}$ celle des $(y_i)_{1\leq i\leq n}$.
Écrire un programme Python permettant de calculer les coefficients a et b à l'aide des données expérimentales récupérées à la question précédente, et des fonctions de la partie 2.

4. On souhaite à présent représenter graphiquement le nuage de points et la droite l'approximant. Nous allons pour ce faire utiliser la bibliothèque `matplotlib.pyplot`.

   (a) Importer cette bibliothèque avec l'instruction `import matplotlib.pyplot as plt`

   (b) Tracer le nuage de points avec l'instruction `plt.scatter(x,y)`.

   (c) L'instruction `plt.plot(L1,L2)` permet de tracer la courbe reliant les points dont les abscisses forment la liste `L1` et les ordonnées forment la liste `L2`. Utiliser cette instruction pour tracer la droite d'équation $y = ax + b$.

   (d) Afficher la fenêtre graphique avec l'instruction `plt.show()`.

5. On donne dans le fichier `'notes.txt'` la liste des notes données par un examinateur à un oral de mathématiques (une note par ligne). Écrire un programme ouvrant ce fichier en lecture et affichant la moyenne de ces notes dans la console.

4 Écriture de fichier

Les fonctions suivantes permettent d'écrire dans un fichier en Python :

- `f=open('nom_du_fichier.txt','w')` permet d'ouvrir un fichier en écriture (*write*), sous le nom de variable `f`. Le fichier sera créé s'il n'existe pas déjà.
- `f.write(ch)`, ajoute dans le fichier `f` ouvert en écriture la chaîne de caractères `ch`. Les caractères `\n` au sein de cette chaîne permettent de passer à la ligne suivante.

Il est toujours nécessaire d'utiliser `close` une fois qu'on a fini d'écrire dans le fichier.

1. Écrire un programme créant un fichier `test.txt` contenant le texte `Hello world!`
2. Pour un coefficient $\gamma \in \mathbb{R}_+^*$ donné, on peut appliquer une *correction gamma* à une note x entre 0 et 20 en remplaçant cette note par $20\times\left(\frac{x}{20}\right)^{\gamma}$. Cela a pour effet de remonter les notes pour $\gamma < 1$, ou de les descendre pour $\gamma > 1$.

   (a) Écrire une fonction `correction` prenant en argument le coefficient `gamma` et la note `x`, et renvoyant la note modifiée, arrondie au dixième supérieur.

   (b) Écrire un programme Python créant et remplissant un fichier `notes2.txt` contenant les notes de `notes.txt` modifiées avec le coefficient $\gamma = 0.8$.

Aide Python :

- Dans la bibliothèque `math` la fonction `ceil(x)` renvoie le plus petit entier supérieur ou égal à `x`.
- Pour transformer un nombre (entier, flottant...) en chaîne de caractères on utilise la fonction `str`.