Les dictionnaires

Les dictionnaires sont des collections similaires aux listes, mais au lieu d'utiliser des index (0, 1, 2, etc…), on utilise des clés alphanumériques ("Poires", "Pommes", "Fraises", etc…).

Les clés sont non ordonnées (c'est-à-dire qu'elles ; ne sont pas forcément rangées dans le même ordre), ainsi lors de l'affichage d'un dictionnaire l'ordre des éléments peut différer d'une exécution à l'autre.

Il faut penser à ne jamais écrire un programme qui prend en compte l'ordre des éléments d'un dictionnaire.

Clés (keys)	Poires	Pommes	Fraises	Eléments (Items)
Valeurs (values)	5	10	35	

Q1. Donner le résultat des exemples suivants

	Script Python	Résultat affiché dans la console

Affichage d'une valeur	`stock={'poires':5,'pommes':10,'fraises':35}`
	`print(len(stock))`
	`print(stock['fraises'])`
		3
		35

Affichage des clés	`stock={'poires':5,'pommes':10,'fraises':35}`
	`inventaire=""`
	`for nom in stock.keys():`
	`  inventaire += nom + ', '`
	`print(inventaire)`
		poires, pommes, fraises,

Affichage des valeurs	`stock={'poires':5,'pommes':10,'fraises':35}`
	`for num in stock.values():`
	`  print (num)`
		5
		10
		35

	Script Python	Résultat affiché dans la console

Affichage des clés et valeurs	`stock = {'poires': 5, 'pommes': 10, 'fraises': 35}`
	
	`for nom, num in stock.items():`
	`  print(nom, '->',num)`
		poires -> 5
		
		pommes -> 10
		
		fraises -> 35

Opérations booléennes	`stock = {'poires': 5, 'pommes': 10, 'fraises': 35}`
	
	`print('fraises' in stock.keys())`
	`print('banane' in stock.keys())`
	`print(7 in stock.values())`
	`print(5 in stock.values())`
	`print(3 in stock.values())`
		True
		
		False
		
		False
		
		True
		
		False

Ajout et edition d'éléments	`stock = {'poires': 5, 'pommes': 10, 'fraises': 35}`
	
	`print(stock)`
	`stock["fraises"]=20`
	`print(stock)`
	`stock["abricots"]=15`
	`print(stock)`
		{'poires': 5, 'pommes': 10, 'fraises': 35}
		
		{'poires': 5, 'pommes': 10, 'fraises': 20}
		
		{'poires': 5, 'pommes': 10, 'fraises': 20, 'abricots': 15}

Suppression d'éléments	`stock = {'poires': 5, 'pommes': 10, 'fraises': 35}`
	
	`print(stock)`
	`del stock["fraises"]`
	`print(stock)`
	`valeur = stock.pop("pommes")`
	`print(stock)`
	`print(valeur)`
		{'poires': 5, 'pommes': 10, 'fraises': 20, 'abricots': 15}
		
		{'poires': 5, 'pommes': 10, 'abricots': 15}
		
		{'poires': 5, 'abricots': 15}
		
		10

Exercice 1 : Entrainement à la manipulation des dictionnaires

Soit le dictionnaire suivant :

```
d = { 'nom':'Dupuis', 'prenom':'Jacque', 'age':30}
```

Q2. Donner le code python qui répond aux questions suivantes :

Corriger l'erreur dans le prénom, la bonne valeur est 'Jacques'.
`d['prenom'] = 'Jacques'`
Afficher la liste des clés du dictionnaire.
`print(d.keys())`
Afficher la liste des valeurs du dictionnaire.
`print(d.values())`
Afficher la liste des paires clé/valeur du dictionnaire.
`print(d.items())`
Ecrire la phrase "Jacques Dupuis a 30 ans.".
`print(d['prenom'], d['nom'], 'a', d['age'], 'ans.')`

Exercice 2 : Ma liste de courses au supermarché

Q3. Construire un dictionnaire D à l'aide de la liste de courses suivante :

Article	Quantité
Croissants	6
Pizza	2
Café en grains (sachet de 500gr)	3
Riz (sachet de 1kg)	1

```
D = {'Croissants': 6, 'Pizza': 2, 'Café en grains': 3,'Riz': 1}
print(D)
```

Q4. Je me suis trompé de quantité pour les pizzas, il faut en ajouter une. Quelle est l'instruction la plus pertinente permettant d'ajouter une pizza dans le dictionnaire ? (Utiliser la bonne clé)

```
D['Pizza'] += 1
print(D)
```

Q5. Utiliser une boucle « `pour` » afin d'afficher les clés(articles) et les valeurs(quantités).

Script Python	Résultat affiché dans la console
`for nom, num in D.items():` `  print(nom, '->',num)`	**Croissants -> 6** **Pizza -> 3** **Café en grains -> 3** **Riz -> 1**

Exercice 3 : Comptage de caractères

On veut réaliser une fonction ***occurrences(chaine)*** qui compte le nombre d'occurrences de chaque caractère d'une chaîne donnée, c'est à dire qu'elle compte le nombre de fois qu'apparait chaque lettre, chiffre, espace, etc… de la chaîne.

Le résultat sera sous la forme d'un dictionnaire où chaque clé sera un caractère de la chaîne et la valeur associée sera le nombre d'occurrences de ce caractère.

Exemple : occurrences('tortue') renvoie `{'t':2, 'o':1, 'r':1, 'u':1, 'e':1}`

Q6. A partir de l'algorithme, donner et tester le script python correspondant.

Algorithme Pseudo-code	Script Python

fonction occurrences(chaine) :
  D est un dictionnaire
  Pour chaque caractère dans la chaine :
  Si caractère est dans les clés de D alors :
  Alors D[caractere] ← D[caractere] +1
  Sinon :
  D[caractère] ← 1
  FinSi
  FinPour
  Renvoie D
Fin
 	
Affiche(occurences('tortue'))
	`def occurrences(chaine) :`
	`  D = {}`
	`  for caractere in chaine :`
	`  if caractere in D.keys() :`
	`  D[caractere] = D[caractere] + 1`
	`  else :`
	`  D[caractere] = 1`
	`  return D`
	
	`print(occurrences('tortue'))`

Définition :

Le terme occurrence indique le nombre de répétitions d'un mot, d'une lettre ou d'une expression dans un texte.

Résultat attendu affiché dans la console
`{'t': 2, 'o': 1, 'r': 1, 'u': 1, 'e': 1}`

Exercice 4 : Anagramme

Définition d'une anagramme : Mot formé en changeant de place les lettres d'un autre mot.
Exemple : Chien, chine, niche.

Q7. En utilisant la fonction occurrence de la question Q6, réaliser une fonction booléenne ***anagramme(chaine1, chaine2)*** qui prend 2 mots en paramètre et renvoie True si *chaine2* est une anagramme de *chaine1*, sinon False.

Script Python	Résultat affiché dans la console
`print(anagramme('tortue', 'tourne' ))` `print(anagramme('imaginer', 'migraine' ))`	**False** **True**

Script Python

```
def anagramme(chaine1, chaine2) :
    occurrenceChaine1 = occurrences(chaine1)
    occurrenceChaine2 = occurrences(chaine2)
    return occurrenceChaine1 == occurrenceChaine2
```

Q8. La fonction anagramme fonctionne-t-elle avec l'exemple suivant ?

```
print(anagramme('le rechauffement climatique', 'ce fuel qui tache le firmament' ))
```

Q9. Pourtant cela est bien une anagramme. Identifier le problème.

Il n'y a pas le même nombre d'espaces dans les deux chaines de caractères.

Q10. Supprimer alors l'élément qui pose problème avant d'effectuer le test des 2 dictionnaires (utiliser la fonction **`del`** montrée en page 2).

Script Python

```
def anagramme(chaine1, chaine2) :
    occurrenceChaine1 = occurrences(chaine1)
    occurrenceChaine2 = occurrences(chaine2)
    del occurrenceChaine1[' ']
    del occurrenceChaine2[' ']
    return occurrenceChaine1 == occurrenceChaine2
```

Exercice 5 : Rendu de monnaie

On veut réaliser une fonction ***renduMonnaie(somme,pieces)*** qui détermine les pièces à rendre dans un monnayeur.

Exemple :

Rendre la somme de 8€	Solution
	1 billet de 5€ 1 pièce de 2€ 1 pièce de 1€

Q10. Traduire l'algorithme en code python

Algorithme pseudo-code	Script Python

Fonction renduMonnaie (somme en entier, pièces : liste des pièces du monnayeur dans l'ordre décroissant) : dictionnaire des pièces choisies
initialiser à zéro le dictionnaire choisies
Pour p dans pieces
choisies[p] ← 0
Tant que somme >= p
somme ← somme-p
choisies[p] ← choisies[p]+1
fin tant que
fin pour
retourner choisies
	`def renduMonnaie(somme,pieces):`
	`  choisies={}`
	`  for p in pieces:`
	`  choisies[p]=0`
	`  while somme>=p:`
	`  somme=somme-p`
	`  choisies[p]+=1`
	`  return choisies`

Script Python	Résultat affiché dans la console

`#pieces en centimes d'euros`
`pieces=[500,200,100,50,20,10,5,2,1]`
`somme=780`
`print('Les pièces choisies sont')`
`print(renduMonnaie(somme,pieces))`
	Les pièces choisies sont
	{500: 1, 200: 1, 100: 0, 50: 1, 20: 1, 10: 1, 5: 0, 2: 0, 1: 0}

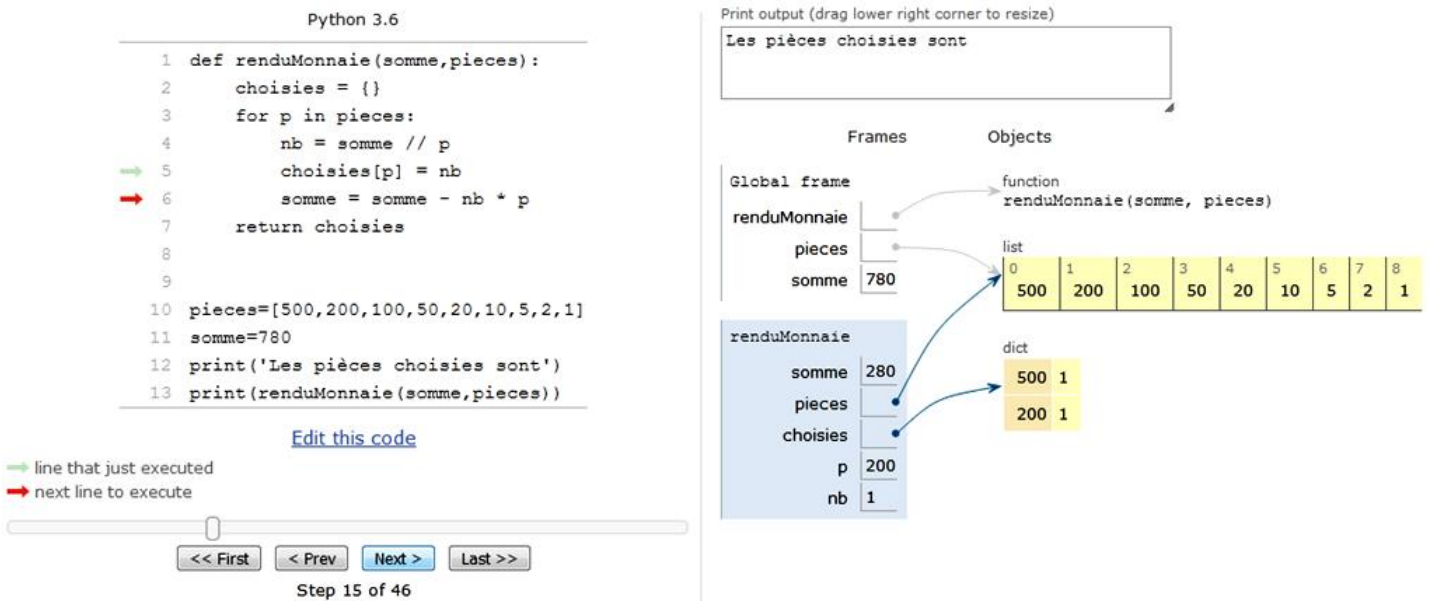
Autre solution :

Algorithme pseudo code	Script Python

Fonction renduMonnaie (somme en entier, pièces : liste des pièces du monnayeur dans l'ordre décroissant) : dictionnaire des pièces choisies
initialiser à zéro le dictionnaire choisie
Pour p dans pieces
nb← somme division entière par p
choisies[p]← nb
somme←somme – nb *p
fin pour
retourner choisies
	`def renduMonnaie(somme,pieces):`
	`  choisies = {}`
	`  for p in pieces:`
	`  nb = somme // p`
	`  choisies[p] = nb`
	`  somme = somme - nb * p`
	`  return choisies`

Rappel : La division entière (on ne garde pas les nombres après la virgule) se fait avec 2 barres obliques //

Visualiser le déroulement du programme dans python tutor



Q11. En termes de performance, lequel des 2 algorithmes précédents est le plus rapide ?

Les 2 algorithmes utilisent une boucle pour, la différence est dans la détermination du nombre de pièces identiques rendues.
Le 1er algorithme utilise une boucle tant que (while) pour trouver le bon nombre identique de pièces.
Le 2ème algorithme utilise un calcul, il n'y a pas de boucle.

Le 2ème algorithme est donc plus rapide.

Exercice 6 : Analyse d'un fichier CSV

Le format CSV (Comma-Separated Values) est un format de fichier texte représentant des données tabulaires dont les valeurs sont séparées par une virgule (par défaut). La première ligne représente les descripteurs, les autres sont les données.

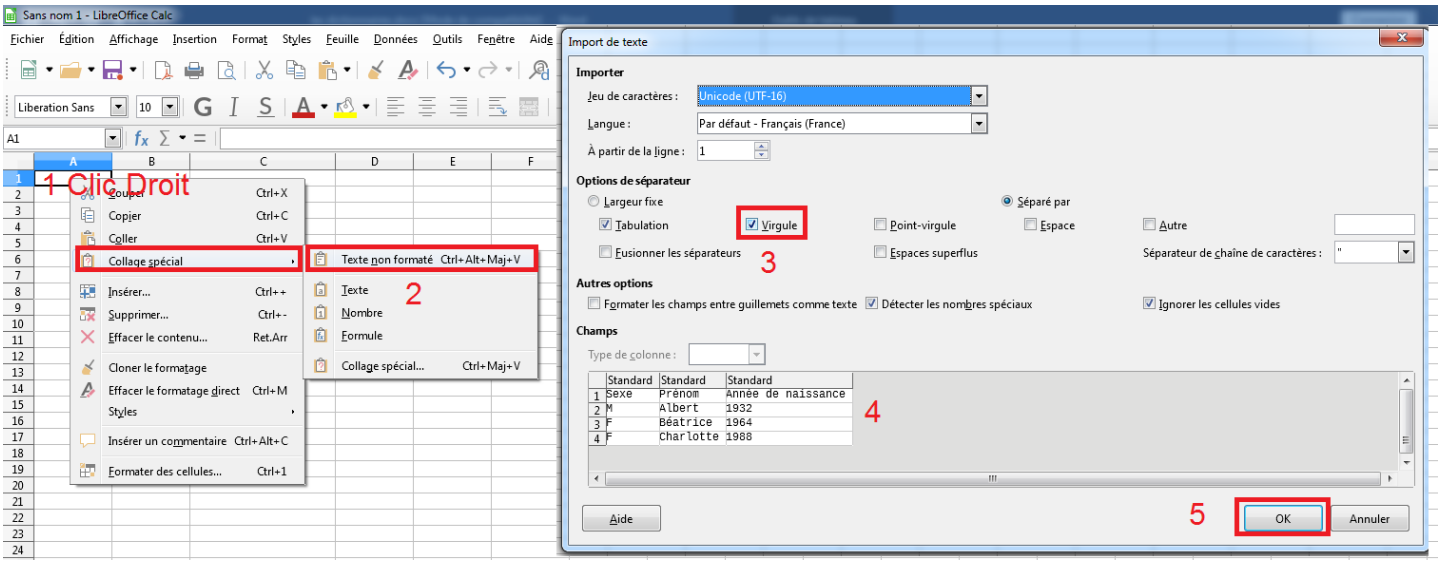
Format .csv	Représentation tabulaire

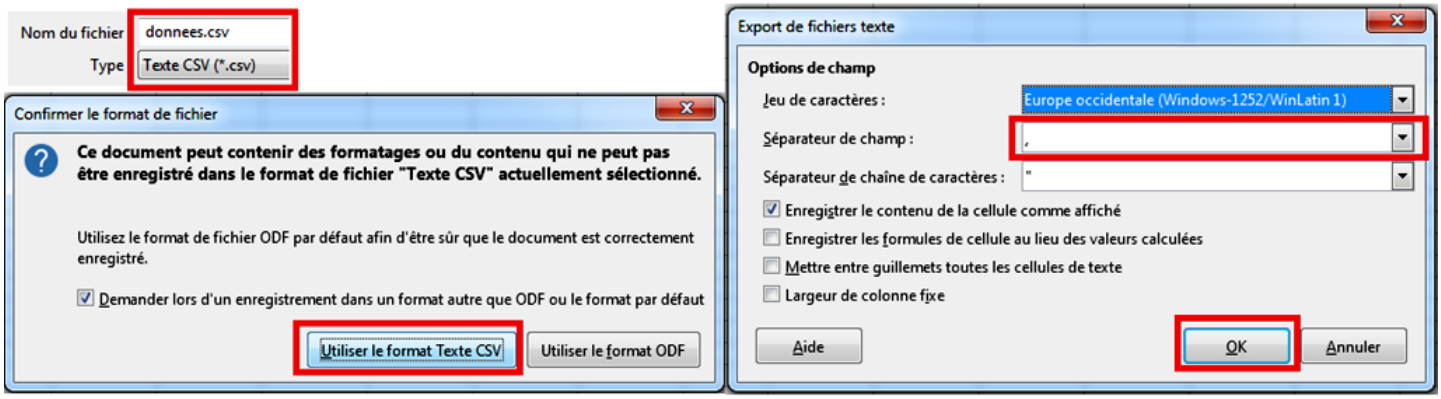
```
Sexe,Prénom,Année de naissance
M,Albert,1932
F,Béatrice,1964
F,Charlotte,1988
```

Sexe	Prénom	Année de naissance
M	Albert	1932
F	Béatrice	1964
F	Charlotte	1988

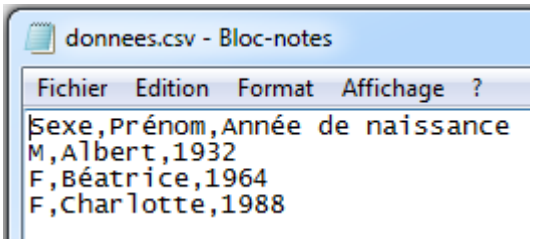
A l'aide du tableur calc effectuer les opérations suivantes :

- Sélectionner le texte ci-dessus (Sexe, prénom, etc..)
- Ouvrir le tableur calc (libre office)
- Cliquer droit sur une cellule (1)
- Coller le texte non formaté (2)
- Choisir la virgule comme séparateur (3)
- Vérifier la conformité de la prévisualisation (4)
- Cliquer sur OK (5)





Vérifier le fichier donnees.csv avec un éditeur de texte comme le bloc-notes



A ce stade, vous devez savoir où se trouve le fichier donnees.csv dans votre disque dur.

Script Python	Explication
`import csv`	Pour lire un fichier CSV, on a besoin d'importer la bibliothèque csv
`fichier = open("donnees.csv", "r")`	Ensuite, on ouvre le fichier en précisant son nom et le mode d'ouverture, ici le mode est lecture « r » car on souhaite lire le fichier.
`lecture_fichier = csv.DictReader(fichier, delimiter=',')`	On lit le contenu du fichier en spécifiant le séparateur (ici ‘,’),

`Liste_personnes = []`
`for ligne in lecture_fichier:`
`  liste_personnes.append(dict(ligne))`
`fichier.close()`
	Enfin, on récupère les données ligne par ligne dans une liste, chaque élément de la liste est un dictionnaire dont les clés sont les descripteurs et les valeurs sont les données du fichier.
	Une fois enregistrées, on ferme le fichier

Script Python : enregistrer sous le nom de fichier lecture_csv.py

```
import csv
fichier = open("donnees.csv", "r")
lecture_fichier = csv.DictReader(fichier, delimiter=',')

liste_personnes = []
for ligne in lecture_fichier:
    liste_personnes.append(dict(ligne))
fichier.close()

for personne in liste_personnes :
    print(personne)
```

Résultat affiché dans la console

```
{'Sexe': 'M', 'Prénom': 'Albert', 'Année de naissance': '1932'}
{'Sexe': 'F', 'Prénom': 'Béatrice', 'Année de naissance': '1964'}
{'Sexe': 'F', 'Prénom': 'Charlotte', 'Année de naissance': '1988'}
```

Cela ne fonctionne pas ?

Les deux fichiers python et csv doivent être dans le **même** répertoire.

Exercice 7 : Analyse d'un fichier météo

Depuis maintenant 10 ans le lycée Touchard lance un ballon sonde stratosphérique.

On souhaite analyser des données météorologiques contenues dans le fichier ballon2019.csv, chaque ligne représente un enregistrement des données des capteurs d'un ballon météo.

Q12. Ouvrir le fichier ballon2019.csv avec le bloc-notes et indiquer comment se présente le caractère séparateur.

```
1  Horaire,Durée,Latitude,Longitude,
2  13:06:50,00:00:00,47.995335,0.204
3  13:07:10,00:00:20,47.995335,0.204
4  13:07:31,00:00:41,47.995335,0.204
5  13:07:52,00:01:02,47.995834,0.205
6  13:08:13,00:01:23,47.9965,0.20616
7  13:08:33,00:01:43,47.996834,0.206
```

Le caractère séparateur est une virgule

Q13. Écrire en premier lieu le programme qui lit ces données et crée une liste de dictionnaires.

Script Python : enregistrer sous le nom de fichier meteo.py

```
import csv
fichier = open("ballon2019.csv", "r", encoding="utf-8")
lecture_fichier = csv.DictReader(fichier, delimiter=',')
liste_enregistrements = []
for ligne in lecture_fichier:
    liste_enregistrements.append(dict(ligne))
fichier.close()
```

Q14. Faire un programme qui affiche le temps de vol (Durée) ainsi que l'altitude (Altitude m).

```
for enregistrement in liste_enregistrements:
    print(enregistrement['Durée'], enregistrement['Altitude m'])
```

Q15. Afficher la moyenne entre la température interne (Temp Int C) et externe (Temp Ext C). Sauvegarder sous le nom meteo2.py

Rappel : La fonction float() permet de changer des caractères alphanumériques en nombre réel (à virgule).

```
for enregistrement in liste_enregistrements:
    print((float(enregistrement['Temp Int C']) + float(enregistrement['Temp Ext C']))/2)
```

Q16 Faire un programme (meteo3.py) qui donne l'heure (Horaire) à laquelle il faisait le plus froid (Temp Ext °C). Vous devez trouver 14:55:18 pour une température de -47.2 °C.

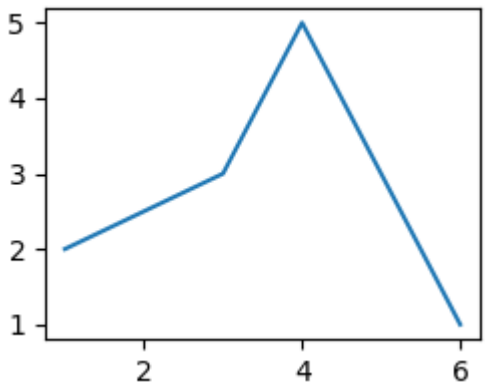
```
horaire_tmp_min = ''
tmp_min = 100
for enregistrement in liste_enregistrements:
    if float(enregistrement['Temp Ext C']) < tmp_min :
        tmp_min = float(enregistrement['Temp Ext C'])
        horaire_tmp_min = enregistrement['Horaire']
print(horaire_tmp_min,tmp_min)
```

Tracé de courbes :

Il n'est pas facile de lire tous ces chiffres, nous allons donc utiliser la bibliothèque Matplotlib, un outil nous permettant d'afficher les données sous forme de graphique.

Exemple de tracé à l'aide de deux listes Y en fonction de X. Le même principe est utilisé en mathématiques pour tracer une courbe à l'aide d'un tableau de valeurs.

	Script Python	Résultat affiché dans la console

Tracé à partir de points	`from pylab import *`
	
	`x = [1, 3, 4, 6]`
	`y = [2, 3, 5, 1]`
	`plot(x, y) # affiche y en fonction de x`
	
	`show() # affiche la figure l'écran`
		

Q17. Dans un nouveau fichier nommé meteo4.py, faire une ***fonction*** **`listePourUneCle(liste_dico,cle)`** qui prend en paramètre une liste de dictionnaires et une clé. Cette fonction retourne un tableau contenant les valeurs de la clé.

Exemple : `listePourUneCle(donnees_meteo, 'Altitude m')` retourne une liste qui contient toutes les altitudes enregistrées par le ballon.

Remarque : les données que l'on souhaite récupérer sont des nombres (float).

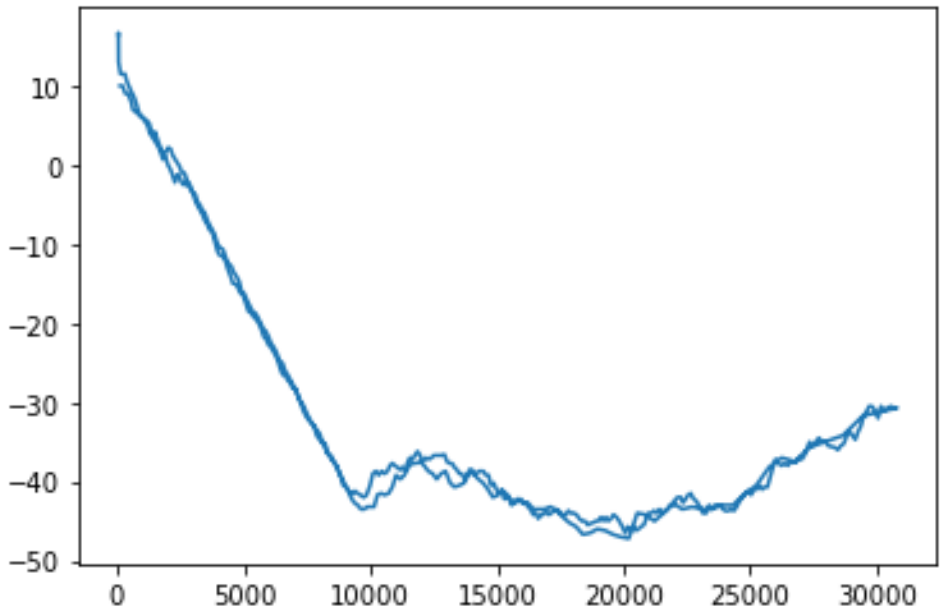
```
import csv
from pylab import *

def listePourUneCle(liste_dico, cle):
    resultat=[]
    for dico in liste_dico:
        resultat.append(float(dico[cle]))
    return resultat


fichier = open("ballon2019.csv", "r", encoding="utf-8")
lecture_fichier = csv.DictReader(fichier, delimiter=',')

liste_enregistrements = []
for ligne in lecture_fichier:
    liste_enregistrements.append(dict(ligne))
fichier.close()

liste_altitudes = listePourUneCle(liste_enregistrements,'Altitude m')
print(liste_altitudes)
```

Q18. A l'aide de la fonction **`listePourUneCle`** de l'exercice précédent et de Matplotlib, compléter le programme ci-dessous qui affiche un graphique qui montre l'évolution de la température en fonction de l'altitude. Donnez le code et une capture d'écran du résultat.

Script python

```
import csv
from pylab import *


def listePourUneCle(liste_dico, cle):
    resultat=[]
    for dico in liste_dico:
        resultat.append(float(dico[cle]))
    return resultat

fichier = open("ballon2019.csv", "r", encoding="utf-8")
lecture_fichier = csv.DictReader(fichier, delimiter=',')

liste_enregistrements = []
for ligne in lecture_fichier:
    liste_enregistrements.append(dict(ligne))
fichier.close()

liste_altitudes = listePourUneCle(liste_enregistrements,'Altitude m')
liste_temperatures = listePourUneCle(liste_enregistrements,'Temp Ext C')

x = liste_altitudes
y = liste_temperatures
plot(x,y)

show()
```

Résultat obtenu



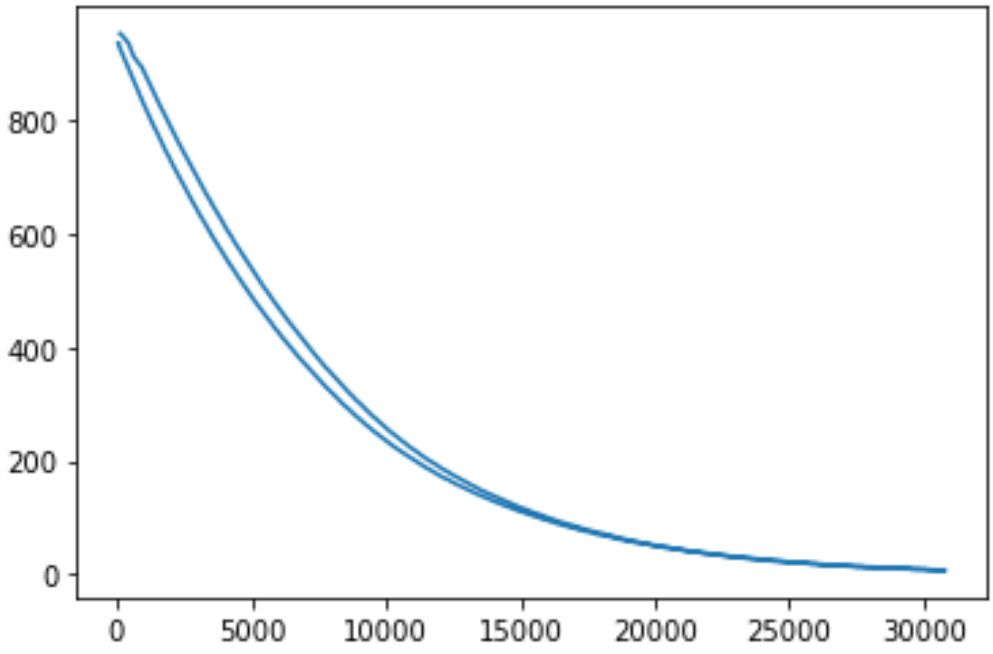
Le graphique semble afficher 2 températures différentes, en effet le ballon est monté jusqu'à une altitude de 30000 mètres, puis il est redescendu.

Q19. Même principe que la question précédente, mais cette fois ci, on désire afficher la pression en fonction de l'altitude.

Script python

```
import csv
from pylab import *


def listePourUneCle(liste_dico, cle):
    resultat=[]
    for dico in liste_dico:
        resultat1.append(float(dico[cle1]))
    return resultat

fichier = open("ballon2019.csv", "r", encoding="utf-8")
lecture_fichier = csv.DictReader(fichier, delimiter=',')
liste_enregistrements = []
for ligne in lecture_fichier:
    liste_enregistrements.append(dict(ligne))
fichier.close()

liste_altitudes = listePourUneCle(liste_enregistrements,'Altitude m')
liste_pressions = listePourUneCle(liste_enregistrements,'Pression mb')


x = liste_altitudes
y = liste_pressions
plot(x,y)

show()
```

Résultat obtenu



La relation entre la pression et l'altitude n'est pas linéaire

Annexe

Création d'un dictionnaire *d*	d = {} d = dict() d = {k1:v1, k2:v2, …}
Accès à la valeur de la clé *key*	value = d[key]
Ajout et édition de la valeur de la clé *key*	d[key] = value
Test d'appartenance de la clé *key* dans *d*	If key in d :
Boucle pour sur chaque clé *key* de *d*	For key in d :
Tests d'égalité	d1==d2 d1 !=d2
Suppression d'un item dont la clé est *key*	del(d[key]) del d[key]

Manipulation des dictionnaires :

Supprime tous les éléments le *d*	d.clear()
Créé une copie de *d*	d.copy()
Créé un dictionnaire à partir d'une liste de clés *ks* et une valeur commune *v*	ks = [k1, k2, …] {}.fromkeys(ks, v)
Liste les éléments de *d*	d.items()
Liste les clés de *d*	d.keys()
Liste les valeurs de *d*	d.values()
Renvoie la valeur de la clé *key* si elle existe, sinon renvoie *value*	d.get(key [, value])
Retire l'item de la clé *key* et renvoie sa valeur Renvoie *value* si *key* n'est pas une clé de *d*	d.pop(key) d.pop(key [, value])
Renvoie la valeur de la clé *key* si elle existe, sinon renvoie *valeur* et créé *d[key] = value*	d.setdefault(key [, value])
Ajoute à *d* la liste de tuples *ts*	d.update(ts)