LES DONNEES DE TYPES CONSTRUITS

1. Les tuples en Python (Activité-1)
2. Les listes en Python (Activité-2)
3. Exercices sur les listes (Activité-3)

4. Les dictionnaires

Comme les listes, les dictionnaires permettent de "stocker" des données. Chaque élément d'un dictionnaire est composé de 2 parties, on parle de paires "**clé/valeur**".

4.1 Premier exemple de dictionnaire :

Saisissez cette ligne de code dans PyScripter et enregistrez le programme sous le nom : « Activite-4_Nom-Prenom »

```
File  Edit  Format  Run  Options  Window  Help
dictionnaire_1 = {"nom": "Dupont", "prenom": "John", "date de naissance": "25/12/2019"}
```

Comme vous pouvez le constater, nous utilisons des accolades { } pour définir le début et la fin du dictionnaire alors que nous utilisons des crochets **[ ]** pour les listes et les parenthèses **( )** pour les tuples.

Dans le **dictionnaire_1** ci-dessus, "nom", "prenom" et "date de naissance" sont des **clés** et "Dupont", "John" et "25/12/2019" sont des **valeurs**.

La clé "nom" est associée à la valeur "Dupont", la clé "prenom" est associée à la valeur "John" et la clé "date de naissance" est associée à la valeur "25/12/2019".

Les clés sont des chaînes de caractères ou des nombres. Les valeurs peuvent être des chaînes de caractères, des nombres, des booléens...

Pour créer un dictionnaire, il est aussi possible de procéder comme suit :

```
File  Edit  Format  Run  Options  Window  Help
dictionnaire_1_bis = {}
dictionnaire_1_bis["nom"] = "Dupont"
dictionnaire_1_bis["prenom"] = "John"
dictionnaire_1_bis["date de naissance"] = "25/12/2019"
```

On peut donc ajouter des paires « clé/valeur » à un dictionnaire existant

Saisissez ces lignes dans PyScripter

Faites afficher les deux dictionnaires en utilisant la commande « print » pour vérifier que les deux dictionnaires sont bien identiques.

Copier-coller votre résultat dans un document word nommé : « Activité-4_Nom-Prénom » que vous compléterez au fur et à mesure de l'activité.

4.2 Quelques fonctionnalités sur les dictionnaires

On peut afficher le contenu du dictionnaire référencé par la variable "dictionnaire" en saisissant " dictionnaire" dans la console.

Faites le test pour les 2 dictionnaires « dictionnaire_1 » et « dictionnaire_1bis »

Il est possible d'afficher la valeur associée à une clé :

Par exemple, on peut demander de faire afficher la phrase : « **Bonjour, je suis John Dupont, je suis né le 25/12/2019**. »

Saisissez ces lignes dans PyScripter et faites compiler votre programme.

```
print ('Bonjour je suis', dictionnaire_1["prenom"] , dictionnaire_1["nom"]+', '+'je suis né le', dictionnaire_1["date de naissance"]+'.')
print (f'Bonjour je suis {dictionnaire_1["prenom"]} {dictionnaire_1["nom"]}, je suis né le {dictionnaire_1["date de naissance"]}.')
```

Commentez la première façon de faire afficher le résultat.

La deuxième façon de faire est le « **formatage de chaîne** » ou « interpolation de chaînes littérales », plus communément appelé « **chaîne f** », en raison du premier caractère « f » précédant la chaîne à afficher.

On peut ajouter un élément à un dictionnaire (les dictionnaires sont mutables)

Ajoutez la clé « ["lieu de naissance"] » dont la valeur est « "Cherbourg" »

Faites alors afficher la phrase : « **Bonjour, je suis John Dupont, je suis né le 25/12/2019 à Cherbourg**. » en utilisant les 2 méthodes vues au dessus.

L'instruction "del" permet du supprimer une paire "clé/valeur"

Quel est le contenu du dictionnaire référencé par la variable "**corbeille_de_fruits**" après l'exécution du programme ci-dessous ?

Vérifiez votre réponse en ajoutant ces lignes de code à votre programme, et en l'exécutant dans la console.

```
corbeille_de_fruits = {"citron": 1, "poire": 3, "pomme": 4, "orange": 2, "banane": 5, "mandarine": 8}
print (corbeille_de_fruits)

del corbeille_de_fruits["orange"]
print (corbeille_de_fruits)
```

On peut aussi modifier une valeur.

Pour modifier la valeur d'une clé d'un dictionnaire, on utilise la commande :

dictionnaire["clé"] = dictionnaire["clé"] – valeur_à_enlever

Complétez votre programme pour retirer 2 pommes du dictionnaire « corbeille_de_fruits ».

On peut aussi déterminer le nombre de valeurs du dictionnaire.

Tester le programme suivant :

```
N=0
for valeur in corbeille_de_fruits:
    N=N+corbeille_de_fruits[valeur]
print('Il y a',N, 'fruits dans la corbeille de fruits.')
print (corbeille_de_fruits)
```

Il est possible de parcourir les clés d'un dictionnaire à l'aide d'une boucle for.

Ce parcours peut se faire selon les clés ou les valeurs. Commençons par parcourir les clés à l'aide de la méthode "keys"

Tester le programme suivant :

```
print("fruits disponibles dans la corbeille :")
for fruit in corbeille_de_fruits.keys():
        print(fruit)
```

La méthode « dictionnaire.values() » permet de parcourir le dictionnaire pour accéder aux valeurs des différentes clés.

Tester le programme suivant :

```
for quantite in corbeille_de_fruits.values():
  print(quantite)
```

Enfin, il est possible de parcourir un dictionnaire à la fois sur les clés et les valeurs en utilisant la méthode « dictionnaire.items() ».

Tester le programme suivant :

```
print ("Stock de fruits :")
for fruit, quantite in corbeille_de_fruits.items():
        print (f"{fruit} : {quantite}")
```

4.3 Exercice d'application : Stock de produits

On travaille sur des stocks ou des commandes de produits . Ces stocks et ces commandes sont gérés en utilisant des dictionnaires. On aura par exemple :

stock={"crayon":43,"stylobleu":23,"stylorouge":40, "stylovert":30, "gomme":12,"regle":25, "compas":10}
commande1={"crayon": 3,"stylovert":23,"gomme":22,"regle":5}
commande2={"crayon": 3,"stylorouge":23,"gomme":22,"regle":5}

1. Écrire une fonction « **nb_objets(commande)**» qui compte le nombre total d'objets d'une commande.

nb_objets (commande1) renvoie :

```
>>> nb_objets(commande1)
53
```

2 . On dispose aussi d'un dictionnaire de prix. Ce dictionnaire est composé de clés qui sont des objets (représentés par des string) et de valeurs (qui sont des float). On peut avoir par exemple :

prix={'regle': 5, 'compas': 6, 'gomme': 1.2, 'crayon': 1.1, 'stylorouge': 3, 'stylobleu': 2.5, 'stylovert': 2.3}

a. Écrire une fonction « **augmente(prix)** » qui augmente tous les prix de 10 %.

augmente(prix) renvoie :

```
>>> augmente(prix)
{'compas': 6.6000000000000005,
 'crayon': 1.2100000000000002,
 'gomme': 1.32,
 'regle': 5.5,
 'stylobleu': 2.75,
 'stylorouge': 3.3000000000000003}
```

b. Écrire une fonction « **facture(commande,prix)** » qui étant donnée une commande et un dictionnaire de prix renvoie le prix total de la commande.

facture(commande2,prix) renvoie :

```
>>> facture(commande2,prix)
123.69999999999999
```

3. Écrire une fonction « **maxi(commande)** »qui renvoie le nom et la quantité d'un des objets dont on a le plus. On renverra un seul nom (n'importe lequel si plusieurs objets réalisent ce maximum).
Il faut au préalable déclarer une variable objet (string) vide : objet="" , et une variable Nmax=0.

maxi(commande1) renvoie :

```
>>> maxi(commande1)
('stylovert', 23)
```

4. Un client passe une commande. Écrire une fonction « **ote(stock,commande)** » qui met à jour le stock. La fonction devra écrire un message si le stock est insuffisant ou si le produit n'existe pas.
exemple:
>>> ote(Stock,commande1) renvoie :

```
>>> ote(stock,commande1)
il n'y a pas de  stylovert  dans le stock
il n'y a pas assez de  gomme , il en manquera  10
Le stock est maintenant:
{'compas': 10,
 'crayon': 40,
 'gomme': 0,
 'regle': 20,
 'stylorouge': 40,
 'stylosbleu': 23}
```

5. Écrire une fonction « **add_commande(commande1,commande2)** » qui additionne deux commandes . On renvoie une commande qui regroupe les deux commandes. (Tous les objets doivent être présents et dans la quantité globale)

Vous devez dans un premier temps définir un dictionnaire « commande » qui est vide et dans lequel il faudra ajouter les différents articles. Commande={}

add_commande(commande1,commande2) renvoie :

```
>>> add_commande(commande1,commande2)
{'crayon': 6, 'gomme': 44, 'regle': 10, 'stylorouge': 23, 'stylovert': 23}
```

4.4 Exercice d'application : dictionnaires de températures

On travaille avec des dictionnaires regroupant des températures.
Un dictionnaire correspondra à une série de relevés dans un certain nombre de villes. La liste des villes est fixe, mais chaque dictionnaire pourra ne contenir qu'un certain nombre de relevés.

listevilles=['caen','paris','londres','moscou','barcelonne','le cap']
dico={'caen':13, 'paris':17, 'barcelonne':31, 'le cap':10}

1. Travail sur un dictionnaire :
   a. Ecrire la fonction **combien(dic)** qui renvoie le nombre de villes présentes dans le dictionnaire dic

```
>>> combien(dico)
4
```

   b. Ecrire la fonction **chauds(dic)** qui renvoie la liste des villes où la température est la plus chaude. On renverra une liste qui pourra contenir les noms d'une ou plusieurs villes.

```
>>> chauds(dico)
['barcelonne']
>>> dico={'paris': 17, 'madrid': 31, 'le cap': 10, 'caen': 13, 'barcelonne': 31}
>>> chauds(dico)
['madrid', 'barcelonne']
```

   c. Ecrire la fonction **saisietemp(lv)** qui étant donnée une liste de villes fait une saisie : le programme propose successivement toutes les villes et demande à l'utilisateur une température pour chacune. Avec ces villes et les températures correspondantes, le programme crée un dictionnaire qui sera renvoyé à la fin de la saisie. Si l'utilisateur ne connait pas la température pour une ville, il tape « N » et la ville ne sera pas mise dans le dictionnaire.

```
>>> saisietemp(listevilles)
veuillez entrer une temperature pour la ville de caen
entrez N si vous ne connaissez pas cette temperature
```

```
veuillez entrer une temperature pour la ville de paris
entrez N si vous ne connaissez pas cette temperature
17
veuillez entrer une temperature pour la ville de londres
entrez N si vous ne connaissez pas cette temperature
N
veuillez entrer une temperature pour la ville de moscou
entrez N si vous ne connaissez pas cette temperature
N
veuillez entrer une temperature pour la ville de barcelonne
entrez N si vous ne connaissez pas cette temperature
31
veuillez entrer une temperature pour la ville de le cap
entrez N si vous ne connaissez pas cette temperature
10
{'barcelonne': 31, 'caen': 13, 'le cap': 10, 'paris': 17}
>>>
```

2. Travail sur une liste de dictionnaires :
On travail maintenant sur une liste de dictionnaires, tous de même style. Cette liste peut contenir un nombre quelconque de dictionnaires.

```
d1={'moscou': 11, 'le cap': 12, 'caen': 12, 'londres': 12, 'barcelonne': 28}
d2={'paris': 10, 'le cap': 26, 'caen': 6}
d3={'paris': 23, 'le cap': 12, 'caen': 18}
l=[d1,d2,d3]
```

a. Ecrire la fonction **nombre(listedico,maville)** qui renvoie le nombre de dictionnaires ayant une clé maville

```
>>> nombre(l,"caen")
3
>>> nombre(l,"londres")
1
>>> nombre(l,"rouen")
0
```

b. Ecrire la fonction **temp(listedico,maville)** qui renvoie la liste des températures pour cette ville en prenant en compte l'ensemble des dictionnaires de la liste.

```
>>> temp(l,"caen")
[12, 6, 18]
>>> temp(l,"rouen")
[]
```

c. Ecrire la fonction **transforme(listedico,listeville)** qui à partir de la liste des dico listedico et de la liste listeville, crée un nouveau dictionnaire dont les clés seront les noms de villes et les valeurs les listes de températures relevées. Si une ville n'a aucune température, elle ne doit pas figurer dans le dictionnaire.

```
>>> transforme(l,listevilles)
{'barcelonne': [28],
 'caen': [12, 6, 18],
 'le cap': [12, 26, 12],
 'londres': [12],
 'moscou': [11],
 'paris': [10, 23]}
```