Travaux pratiques n°3	#Fonctions & Générateurs
Matière : Programmation Python	**Enseignants :** Taoufik Ben Abdallah
Discipline : 1ère année Génie Informatique	Fatma Ben Said
Année Universitaire : 2024-2025 / S1	

Exercice 1

Étant donné la liste des locations des voitures **location** :

```
location=
[['100TR', '16/10/2023', 'V001X', 'V003X', 'V004X'],
 ['100RE', '11/10/2023', 'V001X', 'V002X'],
 ['101KP', '16/11/2023', 'V001X', 'V002X', 'V004X'],
 ['102DD', '19/11/2023', 'V001X', 'V004X']]
voitures={"V001X":"Kia-rio", "V002X":"clio", "V003X":"208", "V004X":"BMW Gran",
"V005X":"Golf5"}
```

La liste **location** est constituée de listes, chacune représentant une **référence**, une **date de location** et **l'identifiant** ou **les identifiants des voitures louées**. Chaque location doit avoir une date unique, c'est-à-dire qu'une seule location peut être créée à la même date.

Le dictionnaire **voitures** représente la base des voitures, où la clé correspond à **l'identifiant de la voiture** et la valeur correspond à sa **marque**.

Soit le programme principal suivant qui appelle les fonctions à définir dans l'exercice :

```
print(get_voiture("V002X"))#{'V002X': 'clio'}
print(get_voitures("V001X","V002X", "V006X"))#[{'V001X':'Kia-rio'},{'V002X':'clio'}]
print(voitures_louees())# {'V001X', 'V002X', 'V003X', 'V004X'}
location_t=transformer_location(location)
voiture_nb=nombre_location(location_t, *voitures_louees())
print(voiture_nb)# {'V002X': 2, 'V003X': 1, 'V004X': 3, 'V001X': 4}
print(liste_marques(**voiture_nb )) # ['clio', 'BMW Gran', 'Kia-rio']
print(liste_marques(5,**voiture_nb )) # None
print(liste_marques()) # None
```

1/ Créer une fonction **get_voiture(id_voiture, v=voitures)** qui prend en paramètres l'identifiant de la voiture à chercher et **v** (par défaut **v=voitures**), et qui retourne sous forme d'un dictionnaire **{'id de voiture':'marque_voiture'}** si elle est trouvée. Sinon, elle renvoie **False**

2/ Créer une fonction **get_voitures(*id_voitures, v=voitures)** qui prend un tuple d'identifiants des voitures **id_voitures** et **v** (par défaut **v=voitures**), et qui retourne une liste de dictionnaires. Chaque dictionnaire a comme **clé l'identifiant de la voiture** et comme **valeur la marque de la voiture**, comme suit : **[{'id_voiture1' : 'desg_voiture1'}, {'id_voiture2': 'desg_voiture2'}, …]**

3/ Créer une fonction **transformer_location(location)** qui prend en paramètres la liste **location** et la transforme sous forme d'un dictionnaire où la **clé** représente **la date de location** et la **valeur** représente un **tuple de dictionnaires des voitures louées**, comme suit :
{'date_location1':({'id_voiture1':'desg_voiture1'},{'id_voiture2': 'desg_voiture2'},…)}

4/ Créer une fonction **voitures_louees(loc=location)** qui prend en paramètres la liste **loc** (par défaut **loc=location**), et qui retourne un ensemble contenant les identifiants des voitures qui sont louées (c-à-d qui se trouvent dans **loc**)

5/ Créer une fonction **`nombre_location(location_t, *v_louees)`** qui prend en paramètre le dictionnaire **`location_t`** et un tuple contenant les identifiants des voitures louées dans **`location_t`**. Elle retourne un dictionnaire ayant pour **clé l'identifiant des voitures** et pour **valeur le nombre de locations** indépendamment de la date associée à chaque voiture

6/ Créer une fonction **`liste_marques(n=2, **v)`** qui prend en paramètre **le nombre minimum de locations des voitures** **`n`** (par défaut **`n=2`**) et le dictionnaire **`v`** qui contient le nombre de locations de chaque voiture indépendamment de la date. Elle **retourne sous forme d'une liste** les marques des voitures à partir de **`v`** qui **sont louées au minimum `n` fois**. Si **`v`** est vide, la fonction retourne **`None`**

Exercice 2

Étant donné la liste des produits **`prod`** :

```
prod=[
 ({'ref':'P001'}, {'Des':'AXRTF'}, {'qte_stock':22}, {'prix_unitaire': 0.9}),
 ({'ref':'P002'}, {'Des':'FRCDR'}, {'qte_stock':6}, {'prix_unitaire': 10}),
 ({'ref':'P003'}, {'Des':'TTRUI'}, {'qte_stock':13}, {'prix_unitaire': 1.1}),
 ({'ref':'P004'}, {'Des':'MLITD'}, {'qte_stock':8}, {'prix_unitaire': 4.6}),
 ({'ref':'P005'}, {'Des':'CAAEP'}, {'qte_stock':0}, {'prix_unitaire': 11}) ]
```

Un produit est caractérisé par sa **référence** **`(ref)`**, sa **description** **`(Des)`**, la **quantité en stock restante** (**`qte_stock)`** et le **prix d'une pièce** **`(prix_unitaire)`**.

Soit le programme principal suivant qui appelle les fonctions à définir dans l'exercice :

```
g=gen_produits()
print(next(g)) #['P001', 'AXRTF', 22, 0.9]
g1=gen_prix()
print(next(g1))#['P001', 'AXRTF', 19.8]
prod_T= transformer_produits1()
print(existe_produit('P003')) #True
print(existe_produit('P006')) #False
transaction(ref="P001", qte=10)
transaction(type="O", ref="P003", qte=3)
```

1/ Écrire un **générateur** **`gen_produits(prod)`** qui prend en paramètre la liste des produits **`prod`** et renvoie toutes les listes possibles correspondantes à un produit de format **`[ref, Des, qte_stock, prix_unitaire]`**

2/ Écrire un **générateur** **`gen_prix()`** qui calcule le prix global de vente de tout le stock de chaque produit de **`prod`** et renvoie toutes les listes possibles correspondantes à un produit de format **`[ref, Des, prix_global]`**

3/ Écrire une fonction **`transformer_produits()`** qui transforme **`prod`** en un tuple de listes de ce format: **`([ref_prod1,Des_prod1,qte_prod1,prix_unitaire_prod1,prix_global_prod1], [ref_prod2,Des_prod2,qte_prod2,prix_unitaire_prod2,prix_global_prod2],…)`** en utilisant le générateur **`gen_produits`**

4/ Écrire une fonction **`existe_produit(ref)`** qui vérifie l'existence d'un produit ayant comme référence **`ref`** dans **`prod_T`**. Cette fonction retourne l'index de la liste contenant le produit dans **`prod_T`** s'il existe et **`-1`** sinon

5/ Écrire une fonction **`transaction(type="I", **p_stock)`** qui permet de mettre à jour le stock de produit **`p_stock`**, représenté par deux arguments `ref` et `qte`, selon **`type`**, qui peut être **`"I"`** (**transaction d'entrée**) ou **`"O"`** (**transaction de sortie)** (par défaut, **`type="I"`**)

Bon Travail ♣