Matière : Programmation Python **Auditoire : PLSI** **Responsable de cours :** Ikram Chaâbane	**Année universitaire 2022-2023**

TP n° 4 : fonctions, gestion des fichiers et des exceptions

Objectifs

1. *Ecrire des fonctions avec un nombre de paramètres fixe et variable*
2. *Gérer des fichiers en Python*
3. *Gérer et déclencher les exceptions dans un code Python*

Exercice 1

Analyser les bouts de code suivants :

Compactage de paramètres (lors de définition de fonction):

```
def moyenne(x, *args):
    somme = x
    for y in args:
        somme += y
    return somme / (1 + len(args))

print(moyenne(5))
print(moyenne(5, 2))
print(moyenne(5, 2, 10, 20, 12, 22))
```

```
def moyenne(x,**kwargs):
    somme = x
    for z,y in kwargs.items():
        somme += y
    return somme / (1+len(kwargs))

print(moyenne(5))
print(moyenne(2,**{"Amine":12,"Ibrahim":14.5}))
print(moyenne(2, Amine= 12, Ibrahim = 14.5))
```

⇨ Le seul paramètre obligatoire est …. On compacte le reste des paramètres dans …….. s'il s'agit de paramètres non nommés et dans …….. s'il s'agit de paramètres nommés.

Décompactage de paramètres (lors de l'appel de fonction):

```
def dire_bonjour(prenom, nom):
    print("Bonjour", prenom, nom)
```

Passage de paramètres nommés

```
d = {"nom": "Gayerie", "prenom": "David"}
dire_bonjour(**d)
```

Passage de paramètres non nommés

```
s = ["David","Gayerie"]
dire_bonjour(*s)
dire_bonjour(*s, *d)
```

Passage de paramètres non nommés et nommés

```
s = ["David"]
d = {"nom": "Gayerie"}
dire_bonjour(*s, **d)
```

Rq. l'opérateur de décompactage * sur un dictionnaire passe seulement les …….. de ce dictionnaire alors que ** passe ……..

Fonction anonyme : lambda

- Création puis exécution

```
f=lambda x : x*x*x;
f(10)
```

- Fonction exécutée lors de création

```
(lambda x : x*x*x)(10)
```

- Fonction qui utilise une fonction anonyme

```
def puissance(n):
    return lambda x : x ** n

puissance(2)
puissance(2)(8)
puissance(8)(2)
```

- Fonction utilise lambda et compactage

```
def puissance(n):
    return lambda *args : sum(args) ** n
puissance(2)(3)
puissance(2)(3,4)
puissance(2)()
```

- Fonction lambda avec map

```
my_list = [1, 2, 3, 4, 5]
print(list(map(lambda x: x > 2, my_list)))
```

Gestion des fichiers et exceptions

```
try:
    file = open("file.txt", "r")
    file.write("Hello World!")
except IOError as e:
    print ("Erreur !!")
    print (e.args, e.strerror)
else:
    print ("Contenu écrit dans le fichier avec succès")
    file.close()
```

```
try:
    file = open("file2.txt", "a")
    file.write("Helloo World!")
finally:
    print ("Malgré tout, je suis toujours exécuté ;)")
    file.close()
```

```
try:
    x=int(input("Saisir un nombre pair différent de 2:\n"))
    assert(x%2==0)
    if x==2:
        raise ValueError()
except AssertionError :
    print(x,"est impair")
except ValueError:
    print("x doit être différent de 2!!")
else:
    with open("file3.txt","a") as f:
        f.write(str(x))
```

Exercice 2

Écrire une fonction pair() qui accepte plusieurs nombres en argument et renvoie une liste avec les nombres pairs. Si on ne passe pas d'argument, elle renvoie None

Exercice 3

1. Écrire une fonction lambda **f** qui accepte trois paramètres **a**, **b**,et **c**. Cette fonction retourne une liste de **a** tuples de deux valeurs val1 et val2 générées aléatoirement avec val1<b et val2<c. Afficher la liste L obtenue.

   NB. L'appel de la fonction crée doit être effectué par un dictionnaire de trois éléments

2. Ouvrir un nouveau fichier **« input.txt »** et écrire quelques lignes où chacune contient 3 nombres séparés par des virgules. Déclencher l'exception **FileNotFoundError** si le fichier n'existe pas.
3. Invoquer **f** sur chaque ligne du fichier « input.txt »

Exercice 4

Écrire une fonction qui retourne 1 si deux ensembles ens1, ens2 de chaines de deux lettres sont complémentaires (comme 'AB' et 'BA'), 0 sinon, et qui déclenche une exception

1. Si les deux ensembles n'ont pas la même longueur
2. Si les deux chaines ne sont pas de deux lettres

Exemple : **ens1**={'AB', 'DA', 'KE'} et **ens2**={'EK', 'BA', 'AD'}
ens1 et **ens2** sont complémentaires, donc la fonction retourne1

Bon travail