Algorithme

Python :scalaire objets

Type:

- **int représente les entiers ex:3**
- **float represente les nombres réels ex:3.14**
- **boolean represente les valeurs True ou False**
- **NoneType speciale et pour valeur None**

```
a=3
print("type de 3: ",type(a))
b=3.14
print("type de 3.14: ",type(b))
b=True
print("type de True: ",type(b))
c=None
print("type de None: ",type(c))
```

```
type de 3:  <class 'int'>
type de 3.14:  <class 'float'>
type de True:  <class 'bool'>
type de None:  <class 'NoneType'>
```

Algorithme

Python :opérations sur les entiers

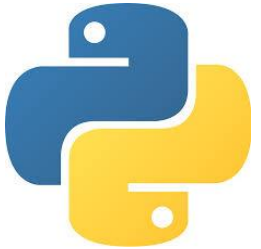
- i+j → la somme le résultat est entier int
- i-j →la différence le résultat est entier int
- i*j →le produit le résultat est entier int
- i/j → la division le résultat est réel float

- Autres opérations
- i%j →le reste de la division entière le résultat est entier int 13%3=1
- i//j → la division entière le résultat est entier int 13//3=4
- i**j → la puissance le résultat est entier int 2**3=8

Python :les chaines de caractère str

```
nom="Ali"
print(nom)
print(type(nom))
# sur multi lignes
info="""le temps
fait beau """
print(info)
#echapement \" permet de taper le caractère "
pays="\"maroc\""
print(pays)
```

```
Ali
<class 'str'>
le temps
fait beau
"maroc"
```



Python :la fonction print

Print():

print:Imprimez un message sur l'écran

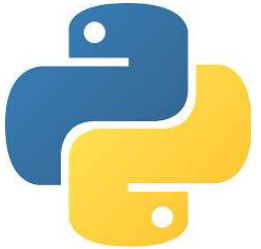
La fonction print () imprime le message spécifié à l'écran, ou tout autre dispositif de sortie standard.

Le message peut être une chaîne, ou tout autre objet, l'objet sera converti en chaîne avant d'être écrit à l'écran

syntaxe:

```
print(chaine1, sep=separator, end=end)
```

chaine1	N'importe quel objet, et autant que vous le souhaitez. Sera converti en chaîne avant d'être imprimé
sep='*separator*'	Optionnel. Spécifiez comment séparer les objets, s'il y en a plus d'un. Par défaut est ' '
end='*end*'	Optionnel. Spécifiez ce qu'il doit imprimer à la fin. Par défaut est '\n'



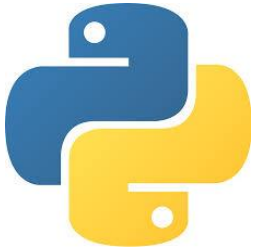
Python :la fonction print

Print():exemple

```
str1="hello"
str2="world"
print(str1,str2,sep="!",end=".\n")
```

```
hello!world.
```

Les deux chaines str1,str2 sont séparées par des ! car le paramétre sep="! ",
A la fin un point et un retour a la ligne car end=".\n")



Python :la fonction input

input()

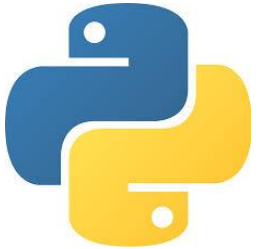
La fonction input () permet à l'utilisateur d'entrer des informations au programme.

Syntaxe:

```
var1=input("message: ") #l'information saisie est retournée en type str(chaine de caractère)
```

```
nom=input("donnez votre nom: ")
age=int(input("donnez votre age: "))
poids=float(input("donnez votre poids: "))
print("salut",nom,"votre age est:",age,"votre p
oids est: ",poids)
```

```
donnez votre nom: RAMI
donnez votre age: 32
donnez votre poids: 65.5
salut RAMI votre age est: 32 votre poids est:  65.5
```



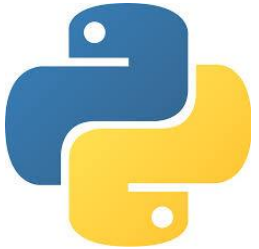
Python :opérations sur les chaines de caractère

concaténation

- L' **+**opérateur utilisé contre deux ou plusieurs chaînes produit une nouvelle chaîne contenant tous les caractères de ses arguments (remarque: l'ordre est important + n'est pas commutative)

```
str1="salut!"
str2=" TDI"
str3=str1+str2
print(str3)
```

```
salut! TDI
```



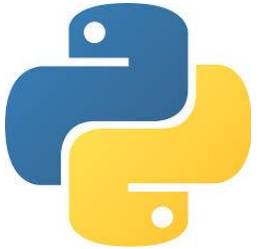
Python :opérations sur les chaines de caractère

Replication:

- l'*opérateur a besoin d'une chaîne et d'un nombre comme arguments; dans ce cas, l'ordre n'a pas d'importance - vous pouvez mettre le nombre avant la chaîne, ou vice versa, le résultat sera le même - une nouvelle chaîne créée par la nième réplication de la chaîne de l'argument

```
str1="a"*5
str2="*"*6
print(str1)
print(str2)
```

```
aaaaa
******
```

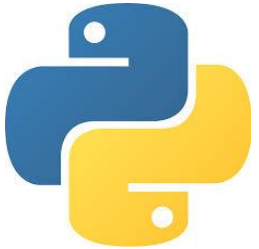


Python : les chaines de caractère Formatage

La méthode format:

```
nom="Rami"
age=33
str1="votre nom est:{} vous avez {} ans".format(nom,age)
print(str1)
```

```
votre nom est:Rami vous avez 33 ans
```



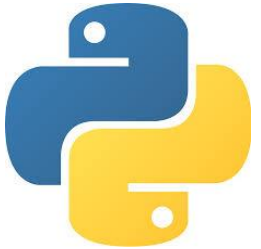
Python : les chaines de caractère Formatage

f-string:

Python f-string est la plus récente syntaxe Python pour faire le formatage des chaines de caractères. Il est disponible depuis Python 3.6. f-string offrent une façon plus rapide, plus lisible, plus concise et moins sujette aux erreurs de formater les chaînes en Python

```
nom="Rami"
age=33
str1=f"votre nom est:{nom} vous avez {age} ans"
print(str1)
```

```
votre nom est:Rami vous avez 33 ans
```



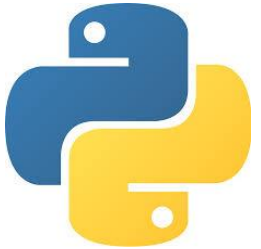
Python : les chaines de caractère Formatage

f-string : avec expression entre{}

avec f-string on peut mettre une expression

```
nom="Rami"
age=33
str1=f"votre nom est:{nom} vous avez {age*12} mois"
print(str1)
```

```
votre nom est:Rami vous avez 396 mois
```

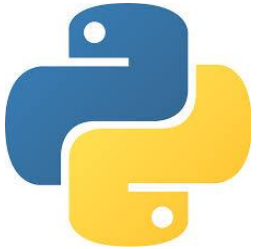


Python : les chaines de caractère Formatage

f-string et debugage: avec expression entre{}

```
nom="Rami"
age=33
str1=f"votre nom est:{nom=} vous avez {age*12=}
 mois"
print(str1)
```

votre nom est:nom='Rami' vous avez age*12=396 mois



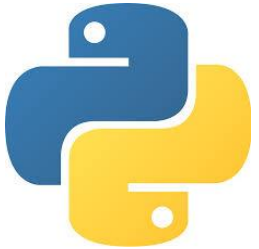
Python : les chaines de caractère Formatage

f-string et dictionnaire

f-string est utilisé pour afficher les informations d'un dictionnaire

```
mydict={"nom":"Rami","prenom":"Ahmed","age":33}
print(f"votre nom complet:{mydict['nom']} {mydict['prenom']} votre age est:{mydict['age']} ans")
```

```
votre nom complet:Rami Ahmed votre age est:33 ans
```



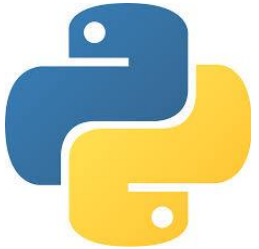
Python : les chaines de caractère Formatage

f-string et caractère d'échapement

f-string est utilisé pour afficher les informations d'un dictionnaire

```
nom="Rami"
prenom="Ahmed"
age=33
print(f"votre nom complet:{nom} {prenom}\nvotre age est:{age} \'ans\'")
```

```
votre nom complet:Rami Ahmed
votre age est:33 'ans'
```



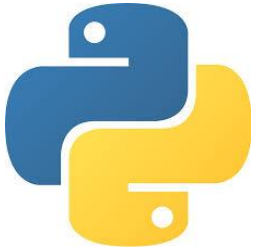
Python : les chaines de caractère Formatage

f-string avec float (réels)

Avec f-string on peut spécifier le nombre de chiffres après la virgule

```
from math import pi
print(f"pi={pi:.4f}")
```

```
pi=3.1416
```



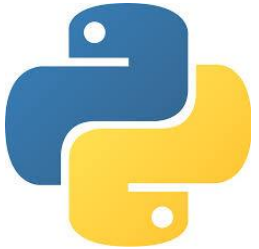
Python : les chaines de caractère Formatage

f-string formatlargeur

Avec f-string on peut spécifier la largeur sur laquelle est formatée la valeur

```
for i in range(1,8):
    print(f"{i:02} {i*i:3} {i**3:4}")
```

```
01  1   1
02  4   8
03  9  27
04 16  64
05 25 125
06 36 216
07 49 343
```



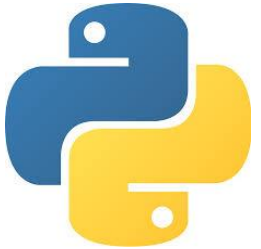
Python : les chaines de caractère Formatage

f-string format numerique notation

Avec f-string on peut spécifier la largeur sur laquelle est formatée la valeur

```
a = 250
#binaire
print(f"{a:b}")
# hexadecimal
print(f"{a:x}")
# octal
print(f"{a:o}")
# scientifique
print(f"{a:e}")
```

```
11111010
fa
372
2.500000e+02
```



Python : les chaines de caractère Formatage

f-string format datetime

```
import datetime

now = datetime.datetime.now()

print(f'{now:%Y-%m-%d %H:%M}')
```

```
2021-03-27 16:13
```

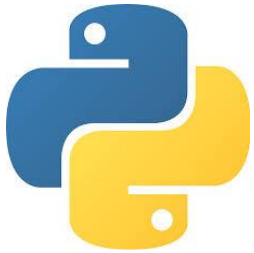
Python : les chaines de caractère Formatage

Les méthodes

Python dispose d'un ensemble de méthodes intégrées que vous pouvez utiliser sur les chaînes.

Toutes les méthodes de chaîne renvoie de nouvelles valeurs. Ils ne changent pas la chaîne d'origine.

Python : les chaines de caractère Formatage

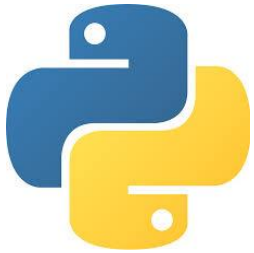


Les méthodes

```
str1="Hello world"
#recuperer la chaine en majuscule
print(str1.upper())
#recuperer la chaine en majuscule
print(str1.lower())
#recuperer la chaine en capitalise
print(str1.capitalize())
#recuperer la chaine avec la premiere lettre de chaque mot en majuscule
print(str1.title())
# retourne la premiére index du mot passé en argument dans str1
print(str1.index("world"))
# retourne la premiére index du mot passé en argument dans str1
 (-1) si le mot n'est pas das str1
print(str1.find("world"))
```

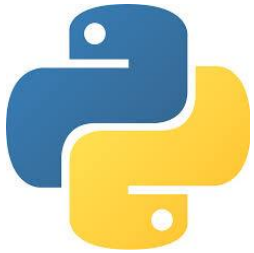
Python : les chaines de caractère Formatage

Les méthodes



Method	Description
capitalize()	Convertit le premier caractère en majus de cas supérieur
casefold()	Convertit la chaîne en miniscule
center()	Retourne la chaine centrée
count()	Renvoie le nombre de fois qu'une valeur spécifiée se produit dans une chaîne
encode()	Renvoie une version codée de la chaîne
endswith()	Retourne vrai si la chaîne se termine avec la valeur spécifiée
expandtabs()	Sets the tab size of the string
find()	Recherche la chaîne pour une valeur spécifiée et renvoie la position de l'endroit où elle a été trouvée
format()	Formats valeurs spécifiées dans une chaîne
format_map()	Formats specified values in a string
index()	Recherche la chaîne pour une valeur spécifiée et renvoie la position de l'endroit où elle a été trouvée

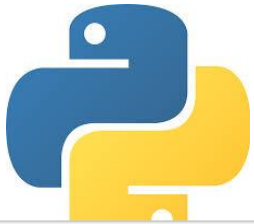
Python : les chaines de caractère Formatage



Les méthodes

isalnum()	Retours Vrai si tous les caractères de la chaîne sont alphanumériques
isalpha()	Retours Vrai si tous les caractères de la chaîne sont alphabets
isdecimal()	Retours Vrai si tous les caractères de la chaîne sont décimales
isdigit()	Retourne vrai si tous les caractères de la chaîne sont des chiffres
isidentifier()	Returns True if the string is an identifier
islower()	Retours Vrai si tous les caractères de la chaîne sont minuscules
isnumeric()	Retours Vrai si tous les caractères de la chaîne sont numériques
isprintable()	Retourne vrai si tous les caractères de la chaîne sont imprimables
isspace()	Retours Vrai si tous les personnages de la chaîne sont des espaces blancs
istitle()	Retourne vrai si la chaîne suit les règles d'un titre
isupper()	Retours Vrai si tous les caractères de la chaîne sont majuscules

Python : les chaines de caractère Formatage

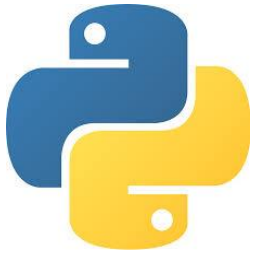


Les méthodes

join()	Joint les éléments d'un itérable jusqu'à la fin de la chaîne
ljust()	Renvoie une version justifiée à gauche de la chaîne
lower()	Convertit une chaîne en minuscule
lstrip()	Renvoie la chaîne en supprimant les espaces en gauche
replace()	Renvoie une chaîne où une valeur spécifiée est remplacée par une valeur spécifiée
rfind()	Recherche la chaîne pour une valeur spécifiée et renvoie la dernière position de l'endroit où elle a été trouvée
rindex()	Recherche la chaîne pour une valeur spécifiée et renvoie la dernière position de l'endroit où elle a été trouvée

Python : les chaines de caractère Formatage

Les méthodes



rstrip()	Renvoie la chaîne sans espace a driote
split()	Divise la chaîne au séparateur spécifié et renvoie une liste
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Renvoie la chaîne sans espace gauche et droite
swapcase()	minuscule devient majuscule et vice versa
title()	Convertit le premier caractère de chaque mot en majuscule
upper()	Convertit une chaîne en majuscule

Logique du contrôle de flux

Pseudo langage : Si alors :

```
X←7;
Si x>10 alors
Ecrire(‘x est superieur strictement à 10’)
Finsi
Ecrire(‘fin’)
```

Python : Si alors :

```
x=7
if x>10:
    print('x est superieur strictement à 10')
print('fin')
```

Logique du contrôle de flux

Pseudo langage : Si alors sinon :

```
X←7;
Si x>10 alors
Ecrire('x est superieur  strictement à 10')
Sinon
Ecrire('x est inferieur ou égal à 10')
Finsi
Ecrire('fin')
```

Python : Si alors sinon :

```
x=7
if x>10:
    print('x est supérieur strictement à 10')
else:
    print('x est inferieur ou égal à 10')
print('fin')
```

Logique du contrôle de flux

Pseudo langage : Si alors sinon si :

```
X←7;
Si x>10 alors
Ecrire('x est superieur strictement à 10')
Sinon si x=10 alors
Ecrire('x est  égal à 10')
Sinon
Ecrire('x est inferieur strictement ')
Finsi
Ecrire('fin')
```

Python : Si alors sinon :

```
x=7
if x>10:
    print('x est superieur à 10')
elif x==10:
    print('x égal à 10')
else:
    print('x est inferieur strictement a 10')
print('fin')
```

Logique du contrôle de flux

EXERCICE 1 : **Si alors sinon si :**

```
X←7;
Si x=1 alors
        Ecrire(‘lundi’)
Sinon si x=2 alors
        Ecrire(‘mardi’)
Sinon si x=3 alors
        Ecrire(‘mercredi’)
Sinon si x=4 alors
        Ecrire(‘jeudi’)
Sinon si x=5 alors
        Ecrire(‘vendredi’)
Sinon si x=6 alors
        Ecrire(‘samedi’)
Sinon si x=7 alors
        Ecrire(‘dimance’)
Sinon
        Ecrire(‘jour invalide’)
Finsi
Ecrire(‘fin’)
```

Python : **Si alors sinon :**

```
x=7
if x==1:
    print('lundi’)
elif x==2:
    print('mardi’)
elif x==3:
    print('mercredi’)
elif x==4:
    print('jeudi’)
elif x==5:
    print('vendredi')
elif x==6:
    print('samedi')
elif x==7:
    print('dimanche')
else:
    print('jour invalide')
print('fin')
```

Logique du contrôle de flux

EXERCICE 2 : **Gestion vente**

Ecrire le programme qui demande à l'utilisateur de donner le nombre d'articles qui il veut acheter.
Le programme affiche le montant à payer
Sachant que le prix de l'article est 100 dh
La quantité en stock est 10
Si le stocke est insuffisant, le programme affiche stock insuffisant sinon le programme demande à L'utilisateur de saisir la valeur de l'argent

Si l'argent est supérieur ou égal au montant
Le programme décrémente le stock puis
Le programme affiche la valeur de l'argent à rendre
Si non
Le programme affiche argent insuffisant
A le fin le programme affiche le message merci pour votre visite

Python : **Si alors sinon :**

```
#Exercice 2 vente
prix=100
stock=10
argent=0
nombre=int(input("donnez le nombre d'articles: "))
print("le montant est: ",prix*nombre)
if nombre<=stock:
    argent=float(input("donnez l'argent: "))
    if argent>nombre*prix:
        stock-=nombre
        print("le reste est:",argent-nombre*prix)
    else:
        print("argent insuffisant")
else:
    print("stock insuffisant")

print("merci pour votre visite")
```

Logique du contrôle de flux instructions répétitives

Pseudo langage : pour :

```
Pour i←1 jusqu'à 10 faire
        ecrire('la valeur de i est : ',i)
finPour
```

Python : for:

```
for i in range(1,11):
    print('la valeur de i est: ',i)
```

```
Pour i←10 jusqu'à 1 faire pas -1
        ecrire('la valeur de i est : ',i)
finPour
```

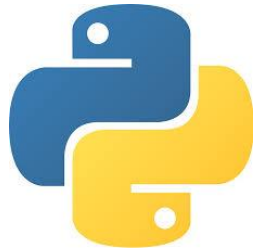
```
for i in range(10,0,-1):
    print('la valeur de i est: ',i)
```

```
range(debut,fin,pas):
```

EXERCICE:Créer une liste contenant les entiers pairs allant de 10 à 20.

```
listPairs=list(range(10,21,2))
```

Logique du contrôle de flux instructions répétitives



Pseudo langage : tantque :

```
i←0
Tantque ( i<=10) faire
        ecrire(‘la valeur de i est : ’,i)
        i←i+1
finTantQue
```

Python : while:

```
i=0
while i <=10 :
    print('la valeur de i est: ’,i)
    i=i+1
```

Les déclarations break and continue dans les boucles while et for

tant que développeur, vous pourriez être confronté aux choix suivants:
• il semble qu'il ne soit pas nécessaire de continuer la boucle dans son ensemble; vous devez vous abstenir de poursuivre l'exécution du corps de la boucle et aller plus loin;
• il semble que vous devez commencer le tour suivant de la boucle sans terminer l'exécution du tour en cours.

Break dans la boucle for :

```
print(" break instruction:")
for i in range(1, 6):
    if i == 3:
        break
    print("dans la boucle for.", i)
print("à l'exterieur de la boucle for.",i)
```

```
 break instruction:
dans la boucle for. 1
dans la boucle for. 2
à l'exterieur de la boucle for. 3
```

Continue dans la boucle for:

```
# continue - example
print("\n  continue instruction:")
for i in range(1, 6):
    if i == 3:
        continue
    print("dans la boucle for.", i)
print(" à l'exterieur de la boucle for.",i)
```

```
  continue instruction:
dans la boucle for. 1
dans la boucle for. 2
dans la boucle for. 4
dans la boucle for. 5
 à l'exterieur de la boucle for. 5
```

Les déclarations break and continue dans les boucles while et for

Break dans la boucle while :

```
print(" break instruction:")
i=1
while i<=5:
    if i == 3:
        break
    i=i+1
    print("dans la boucle while.", i)
print("à l'exterieur de la boucle while.",i)
```

```
 break instruction:
dans la boucle for. 1
dans la boucle for. 2
à l'exterieur de la boucle for. 3
```

Continue dans la boucle while:

```
i=0
while i<5:
    i=i+1
    if i == 3:
        continue
    print("dans la boucle while.", i)
print(" à l'exterieur de la boucle while.",i)
```

```
  continue instruction:
dans la boucle for. 1
dans la boucle for. 2
dans la boucle for. 4
dans la boucle for. 5
 à l'exterieur de la boucle for. 5
```

Exercice d’application break and continue

Exercice mangeur de voyelles:

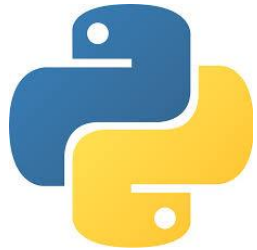
Votre tâche ici est très spéciale : vous devez concevoir un voyelle!
Écrivez un programme qui utilise:

- une boucle **for** ;
- le concept d'exécution conditionnelle (if-elif-else)
- le mot clé **continue** déclaration.

Votre programme doit:

- demander à l'utilisateur de saisir un mot;
- utiliser userWord = userWord.upper() pour convertir le mot entré par l'utilisateur en majuscules; nous parlerons des méthodes dites de chaîne et de la upper() méthode très bientôt
- utiliser l'exécution conditionnelle et **continue** instruction pour "manger" c-à-dire supprimer les voyelles suivantes A , E , I , O , U du mot entré;
- imprimer les lettres non consommées à l'écran, chacune d'elles sur une ligne distincte. Testez votre programme avec les données que nous vous avons fournies.

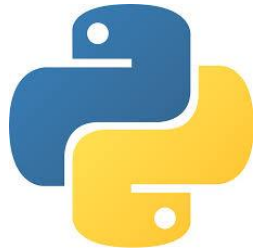
Exercice d’application break and continue



Exercice mangeur de voyelles:CORRECTION

```
#mot=input('donnez un mot')
mot='gregory'
mot=mot.upper()
#les voyelles sont A , E , I , O , U
for lettre in mot:
    if lettre in 'AEIOU':
        continue
    print(lettre)
```

Exercice d’application break and continue



Exercice mangeur de voyelles:CORRECTION

Apporter des améliorations au programme :

créer wordWithoutVovelset attribué une chaîne vide. Utilisez l'opération de concaténation pour demander à Python de combiner les lettres sélectionnées en une chaîne plus longue lors des tours de boucle suivants, et affectez-la à la wordWithoutVovels variable. Testez votre programme avec les données que nous vous avons fournies.

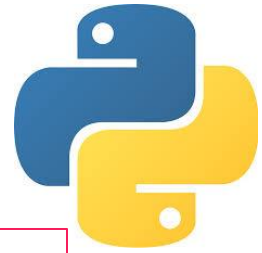
```
#les voyelles sont A , E , I , O , U
wordWithoutVoyels=''
for lettre in mot:
    if lettre in 'AEIOU':
        continue
    wordWithoutVoyels=wordWithoutVoyels+lettre
print(wordWithoutVoyels)
```

La boucle while et for et la branche else

Exemple

```
#while et branche else
i = 1
while i < 5:
    print(i)
    i += 1

else:
    print("else while:", i)

#for et branche else
for i in range(1, 5):
    print(i)
else:
    print("else for:", i)
```

Fonction



Définition fonction:

Une fonction est un bloc de code qui ne s'exécute que lorsqu'il est appelé.
Vous pouvez transmettre des données, connues sous le nom de paramètres, dans une fonction.
Une fonction peut retourner des données en conséquence.
En python une fonction est définit en utilisant le mot clé **def**

Pour appeler une fonction utiliser le nom de la fonction suivi de parenthèses et les valeurs des paramètres (arguments)

Fonction

Création & appel de fonction:

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

#appel de la fonction salutation
nom='Alami'
prenom='Ahmed'
salutation(nom,prenom)
#un autre appel de la fonction salutation
salutation('Fatihi','Khalid')
```

```
Bonjour Alami Ahmed
Bonjour Fatihi Khalid
```

On dit que la fonction salutation a deux paramètres nom et prenom
Ou bien la fonction salutation reçoi comme argument le nom et le prenom
Cette fonction fait un traitement mais elle ne retourne rien.

```
print(type(salutation('Fatihi','Khalid')))
```

```
<class 'NoneType'>
```

Un paramètre est la variable énumérée à l'intérieur des parenthèses dans la définition de la fonction.
Un argument est la valeur qui est envoyée à la fonction lorsqu'elle est appelée.

Fonction

appel de fonction par arguments positionnels:

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

```
#appel de la fonction salutation
n='Alami'
p='Ahmed'
salutation(n,p)
```

```
#un autre appel de la fonction salutation
salutation('Fatihi','Khalid')
```

Fonction

Nombre d'arguments:

Par défaut, une fonction doit être appelée avec le bon nombre d'arguments. Ce qui signifie que si votre fonction s'attend à 2 arguments, vous devez appeler la fonction avec 2 arguments, pas plus, et pas moins.

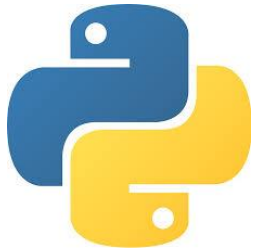
```
def salutation(nom,prenom):
    print("Bonjour",nom,prenom)
#appel correcte
salutation('Fatihi','khalid')
#appel faux car 3 arguements
salutation('Fatihi','khalid','brahim')
```

```
TypeError: salutation() takes 2 positional arguments but 3 were given
```

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

#appel faux car 1 arguement
salutation('Fatihi')
```

```
TypeError: salutation() missing 1 required positional argument: 'prenom'
```

Fonction



appel d'une fonction avec Arguments par mots clés

Vous pouvez également envoyer des arguments avec
clé = valeur.
De cette façon, l'ordre des arguments n'a pas d'importance.

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

salutation(nom='Rami',prenom='Ahmed')
salutation(prenom='Ahmed',nom='Rami')
```

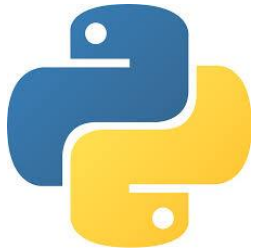
Fonction

Valeur de paramétré par défaut

L'exemple suivant montre comment utiliser une valeur de paramètre par défaut.
Si nous appelons la fonction sans argument, elle utilise la valeur par défaut :

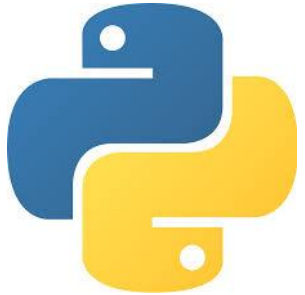
```
def presenter(nom,prenom,pays='Maroc'):
    print("Je suis ",nom,prenom,' du ',pays)

#premier appel
presenter("Alami",'Ahmed’)
#autre appel
presenter("Michelle",'patrique','France')
#autre appel
presenter(prenom="patrique",pays='France',nom='Michelle')
#autre appel
presenter('Fahmi',prenom="patrique"'',pays='Tunisi')
#autre appel avec erreur
presenter(prenom="patrique",pays='Tunisi')
```

Fonction



Exercice

donner la fonction qui retourne la factorielle n! d'un nombre entier donnée



Structures in pyton

La structure de données la plus basique de Python est la séquence. Chaque élément d'une séquence se voit attribuer un nombre - sa position ou son index. Le premier indice est zéro, le deuxième index en est un, et ainsi de suite. Python a six types intégrés de séquences, mais les plus courantes sont des listes et des tuples, que nous verrions dans ce cours. Il y a certaines choses que vous pouvez faire avec tous les types de séquences. Ces opérations comprennent l'indexation, le tranchage, l'ajout, la multiplication et la vérification de l'adhésion. En outre, Python a des fonctions intégrées pour trouver la longueur d'une séquence et pour trouver ses éléments les plus grands et les plus petits.

List in pyton

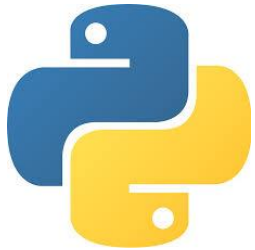
La liste est le type de données le plus utilisé dans Python, qui peut être écrit comme une liste de valeurs séparées par virgule (éléments) . Ce qui est important dans une liste, c'est que les éléments d'une liste ne doivent pas être du même type. Créer une liste est aussi simple que de mettre différentes valeurs séparées par virgule

```
list1 = ['math', 'physique', 10,True, 2020]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

Accéder aux valeurs dans les listes

Pour accéder aux valeurs dans les listes, utilisez les crochets pour trancher avec l'index ou les indices pour obtenir la valeur qui correspond à cet indice

Tableaux in Python (List)



C’est quoi une Liste ?

What is an Array?

Une liste est une variable spéciale, qui peut contenir plus d’une valeur à la fois.

Si vous avez une liste d’éléments (une liste de noms de fruits, par exemple), la liste stocke les fruits en variables simples la liste pourrait ressembler à ceci :

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
```

Tableaux in Python (List)

List

on peut utiliser les listes comme tableaux, cependant, pour travailler avec des tableaux dans Python, vous devrez importer une bibliothèque, comme la bibliothèque NumPy.

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
notes=[12,14,10,8,14,17,9,13.5]
```

Balayer les éléments de la liste

```
#balayer les elements d'une liste
for fruit in fruits:
    print(fruit)
print('_'*20)
for f in fruits:
    print(f)
```

Tableaux in Python (List)

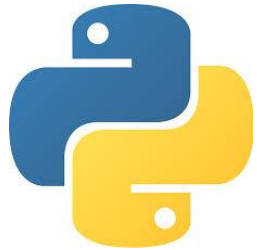
Longueur d'une Liste len

la fonction python **len** qui reçoit comme arguement une liste retourne la longueur de la liste
len peut aussi reçoit comme argument une chaine de caractère et retourne sa longueur

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
notes=[12,14,10,8,14,17,9,13.5]
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
notes=[12,14,10,8,14,17,9,13.5]
print(fruits)  #retourne['orange','pomme','fraise','raisin','poire','kiwi']
print(type(fruits)) #retourne <class,'list'>
print(len(fruits))#retourne 6
print(len(notes))#retourne 8
str1="salut tdi"
print(len(str1)) #retourne 9
```

Pour définir une liste vide
myListe=[]
#cette liste est vide sa longueur est 0

Tableaux in Python (List)



Les mot clé in et not in

le mot clé **in** permet de verifier si une valeur appartient aux éléments de la liste
'pomme' in fruits retourne True si 'pomme' est dans la liste fruits

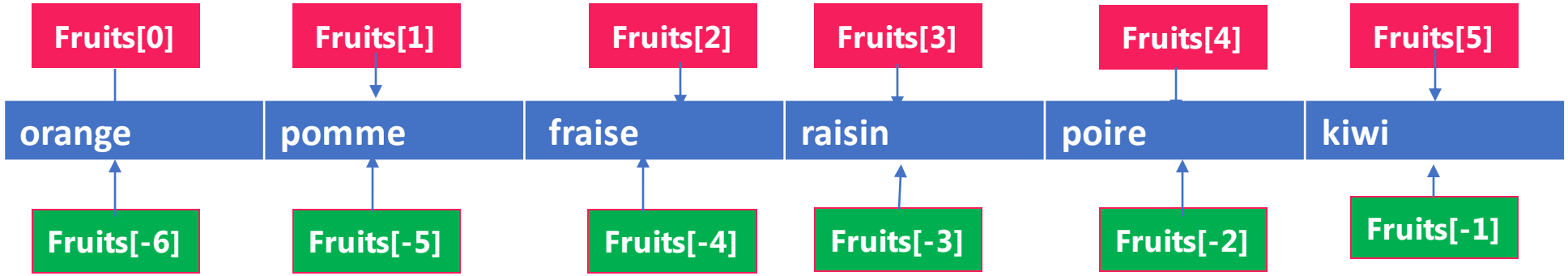
```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']

#mot clé in  pomme se trouve dans la liste?
existFruit='pomme' in fruits
print(existFruit)
```

Tableaux in Python (List)

Accéder aux éléments d'une liste

un élément dans une liste est référencé par son indice (index)
le premier élément a pour indice zéro

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
f1=fruits[0]
print(f1) #affiche orange
print(fruits[2]) #affiche fraise
n=len(fruits) # n contient la longueur de la liste 6
print(fruits[n-1]) #affiche le dernier élément kiwi
print(fruits[-1]) #affiche le dernier élément kiwi
```



Tableaux in Python (List)

Modifier les éléments d'une liste

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
fruits[0]='banane'
fruits[1]=fruits[-1]
print(fruits)
```

```
['banane', 'kiwi', 'fraise', 'raisin', 'poire', 'kiwi']
```

Tableaux in Python (List)

Ajouter un élément à une liste

Vous pouvez utiliser la méthode **append()** pour ajouter un élément à un tableau.

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
fruits.append('mango')
print(fruits)
```

```
['orange', 'pomme', 'fraise', 'raisin', 'poire', 'kiwi', 'mango']
```

Tableaux in Python (List)

Supprimer un élément d'une liste

La méthode **pop()** permet de supprimer un élélment
La méthode pop() génére une exception si la liste est vide ou l'index est hors de l'intervalle

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
fruitDeleted=fruits.pop()
print(fruits)
print("fruit supprimé: ",fruitDeleted)
#supprimer un élément a une position
fruits.pop(1)
print(fruits)
#supppression par l'instruction del
del fruits[2]
print(fruits)
```

```
['orange', 'pomme', 'fraise', 'raisin', 'poire']
fruit supprimé:  kiwi
['orange', 'fraise', 'raisin', 'poire']
['orange', 'fraise', 'poire']
```

Tableaux in Python (List)

Supprimer un élément par l'expression del

On peut supprimer u élément par l'expression del

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
del fruits[2] #supprime l'élément qui a pour indice 2 'fraise'
print(fruits)
```

```
['orange', 'pomme', 'raisin', 'poire', 'kiwi']
```

Tableaux in Python (List)

inserer un élément a une position donnée dans liste

La méthode insert(position,objet) permet d'insérer un élément a une postion

```
fruits=['orange','pomme','fraise','raisin','poire','kiwi']
fruits.insert(2,'banane')
print(fruits)
```

```
['orange', 'pomme', 'banane', 'fraise', 'raisin', 'poire', 'kiwi']
```

Tableaux in Python (List)

Queleque fonction pour liste

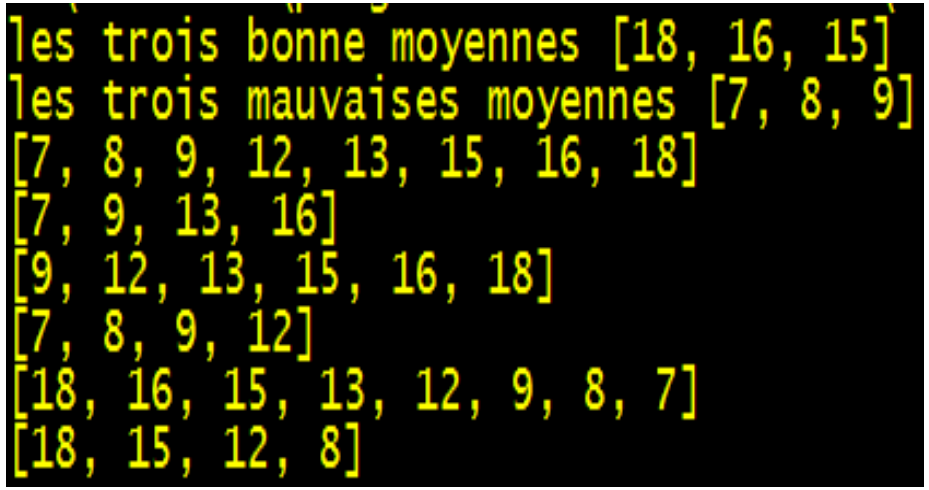
append()	ajouter un élément a la fin d'une liste
clear()	Supprimer tous les élément d'une lliste
copy()	Retourne une copie d'une liste
count()	retourne le nombre d'élément passé en argument exemple **fruits.count('pomme')**
extend()	Ajoute les éléments d'une liste,à la fin de la liste en cours fruits.extend(['melon','mango'])
index()	Retourne l'index du premier élément equivalent a la valeur passé en argument
insert()	Ajouter un élément a une position `fruits.insert(2,'banane')`
pop()	Supprimer l'élément equivalent à la position donnée
remove()	Removes the first item with the specified value supprime le premier élément equivalent à la valeur passé en argument
reverse()	renverse l'ordre de la liste
sort()	Tri la liste

Tableaux in Python (List)

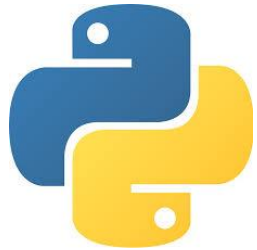
Tranches slices

Exercice 1:
On peut récupérer des Tranches d'une liste par
malist[debut,fin,pas]

```
list1=[7,8,9,12,13,15,16,18]
list1.sort()
print("les trois bonne moyennes",list1[-1:-4:-1])
print("les trois mauvaises moyennes",list1[0:3])

print(list1[:])#[7,8,9,12,13,15,16,18]
print(list1[2:]) #[9,12,13,15,16,18]
print(list1[:4]) #[7,8,9,12]
print(list1[-1::-1])#[18,16,15,13,12,9,8,7]
print(list1[-1::-2])  #[18,15,12,8]
```



Tableaux in Python (List)



Exercices fonction & liste

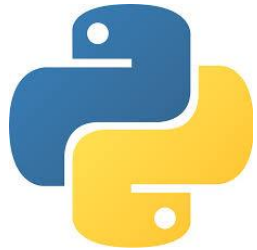
Exercice 1:

Écrire une fonction `elimine(L)` qui supprime les doublons d'une liste:

`elimineDoublant``([1,2,1,3,1,4,2,2,1,1])` renvoie `[1,2,3,4]`.

```
def elimineDoublant(myList):
    res=[]
    for el in myList:
        if el not in res:
            res.append(el)
    return res


liste1=[1,2,1,3,1,4,2,2,1,1]
print(elimineDoublant(liste1))
```

Tableaux in Python (List)

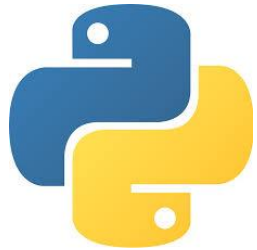


List Comprehension

List comprehension
offre une syntaxe plus courte lorsque vous souhaitez créer une nouvelle liste basée sur les valeurs d'une liste existante.

valeur	collection	condition
List2= [x*2	For x in range(0,10)	**If x%2==0]**

Tableaux in Python (List)



List Comrehension

valeur	collection	condition
List2= [x*2	For x in range(0,10)	If x%2==0]

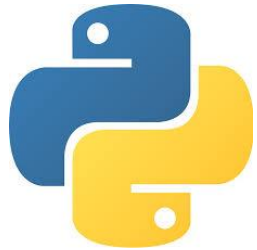
Exercice 1:

Donner le programme Python qui permet de créer une liste contenant les nombres paires qui qui sont inferieur a 100

```
#methode classique
print('création par list par methode classique')
list2=[]
for i in range(0,50):
    if i%2==0:
        list2.append(i)
print(list2)
```

```
#list par list comprehension
print(' création par list comprehension ')
list1=[i for i in range(0,50) if i%2==0]
print(list1)
```

Tableaux in Python (List)



List Comrehension Exercice

valeur	collection	condition
List2= [x*2	For x in range(0,10)	If x%2==0]

List comprehension Exemple:

Donner la liste relative a l'expression comprehension suivante:

```
mesReste=[(x**2)%9 for x in range(1,21)]
print (mesReste)
```

Tableaux in Python (List)

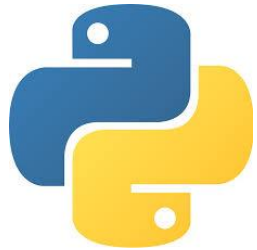
List of tuple Comrehension

valeur	collection	condition
List2= [x*2	For x in range(0,10)	If x%2==0]

List of tuple comprehension Exemple:

```
mylist=[(1,2),(3,4),(5,6)]
mylist1=[(x,x*2) for x in range(6)]
print(mylist1)
mylist2=[x*y for x,y in mylist]
print(mylist2)
```

Tableaux in Python (List)



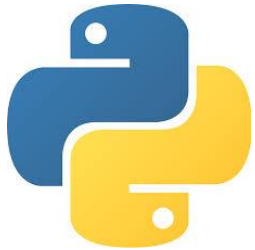
Exercices fonction & liste

Exercice 2:

- Écrire un programme qui demande à l'utilisateur de saisir une liste d'entiers, puis à l'aide de parcours successifs de la liste effectuer les actions suivantes :

  1. Afficher la liste
  2. Afficher la liste en colonne de manière à afficher l'index et sont contenu
  3. Additionner tous les éléments de la liste.
  4. Créer une nouvelle liste qui sera le multiple **(3)** de tous les éléments de la liste.
  5. Obtenir le plus grand nombre de la liste.
  6. Obtenir le plus petit nombre de la liste.
  7. Compter le nombre des nombres pairs présents dans la liste
  8. Calculer la somme de tous les nombres impairs de la liste

Tableaux in Python (List)



Exercices fonction & liste

Exercice 3:

- La liste suivante représenta les moyennes d'une classe
- **moyennes=[14.84,14.14,16.22,86,85,85,14.84,13,15.85,9.99,12.04,15.03,16.22,12,84,10.20,11.03,11.03]**
  - Afficher les trois bonnes moyennes
  - Afficher les trois mauvaises moyennes (triées de plus petites au plus grandes)

Tableaux in Python (List)

Exercices fonction & liste

Exercice 4:

- Vous trouvez des erreurs lors de l'évaluation du code suivant. Les numéros de ligne sont inclus pour référence seulement.

```
1 numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
2 index = 0
3 while (index < 10)
4     print (numbers [index])
5
6     if numbers(index) = 6:
7         break
8     else :
9         index += 1
```

Tableaux in Python (List)

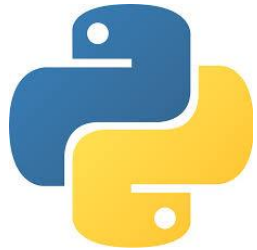
Exercices fonction & liste

Exercice 5:

- Définir la liste suivante: `liste =[17, 38, 10, 25, 72]`, puis effectuez les actions suivantes :

1. Triez et affichez la liste ;
2. Ajoutez l'élément 12 à la liste et affichez la liste ;
3. Renversez et affichez la liste ;
4. Affichez l'indice de l'élément 17 ;
5. Enlevez l'élément 38 et affichez la liste ;
6. Affichez la sous-liste du 2e au 3e élément ;
7. Affichez la sous-liste du début au 2e élément ;
8. Affichez la sous-liste du 3e élément à la fin de la liste ;
9. Affichez la sous-liste complète de la liste ;
10. Affichez le dernier élément en utilisant un indiçage négatif

Structure en python (dictionnaire)



Définition

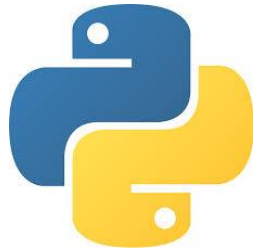
Un **dictionnaire** en Python va permet de rassembler des éléments mais ceux-ci seront identifiés par une **clé**. On peut faire l'analogie avec un dictionnaire de français où on accède à une définition avec un mot.
Contrairement aux listes qui sont délimitées par des crochets, on utilise des **accolades** pour les dictionnaires.

Un élément a été défini ci-dessus dans le dictionnaire en précisant une clé au moyen d'une chaîne de caractères suivie de : puis de la valeur associée
clé: valeur

On accède à une valeur du dictionnaire en utilisant la clé entourée par des **crochets** avec la syntaxe suivante :

```
stock={"article1":120,"article2":200,"article3":50}
n1=stock["article1"]
print(n1)
```

Structure en python (dictionnaire)



Accéder à une valeur:

```
# recupération par clé
stock={"article1":120,"article2":200,"article3":50}
n1=stock["article1"]
print(n1)
#recuperation par get

n=stock.get('article1')
```

Len(stock) retourne la longueur du dictionnaire

Structure en python (dictionnaire)

Ajouter une clé:valeur au dictionnaire:

```
stock={"article1":120,"article2":200,"article3":50}
stock["article4"]=100
stock["article1"]=125

print(stock)
```

```
{'article1': 125, 'article2': 200, 'article3': 50, 'article4': 100}
```

Remarque :

il faut noter que les clés dans un dictionnaire sont uniques

L'instruction:

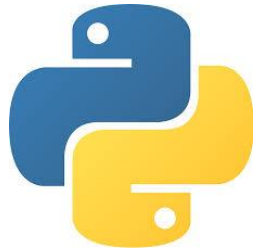
```
stock["article4"]=100   ajoute une clé 'article4'
```

Remarquez que

L'instruction

```
stock["article1"]=125 met à jour la valeur de la clé 'article1' car il
existe déjà dans le dictionnaire
```

Structure en python (dictionnaire)

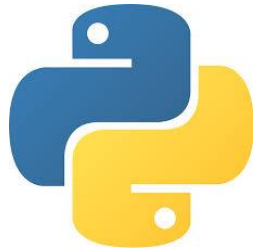


Supprimer un élément d'un dictionnaire:

La suppression peut se faire par l'instruction **del** ou bien par la méthode **pop()**

```
inventaire={'orange':120,'pomme':100,'fraise':50,'raisin':40}
print(inventaire)
#supprimer un elm par l'instruction del
print('supprimer pomme')
del inventaire['pomme']
#génére une erreur si la clé n'est pas dans le dictionnaire
print(inventaire)
#supprimer un elm par la méthode pop
print('supprimer raisin')
n=inventaire.pop('raisin') # retourne la valeur correspond au clé supprimé 40
print(inventaire)
#pop genere une erreur si la clé n'est pas dans le dictionnaire
# pour ne pas avoir d'erreur si la clé n'est pas dans le dictionnaire
#ajouter None comme deuxiéme argument
inventaire.pop('kiwi',None)
```

Structure en python (dictionnaire)



Balayer un dictionnaire:

```
Balayer les clé:
for k in stock.keys():
    print(k)
Balayer les valeurs:
for v in stock.values():
    print(v)
Balayer clé/valeur:
for k,v in stock.items():
    print(k,v)
```

Structure en python (dictionnaire)

La methode update dictionnaire:

La méthode update permet de faire la mise à jour d'un dictionnaire à partir d'un autre dictionnaire

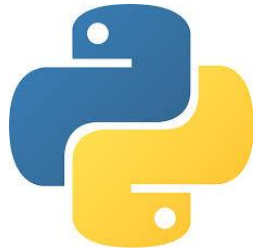
```
inventaire={'orange':120,'pomme':100,'fraise':50,'raisin':40}
inventaire2={'orange':20,'fraise':45,'raisin':25,'kiwi':34}
inventaire.update(inventaire2)
print(inventaire)
```

```
{'orange': 20, 'pomme': 100, 'fraise': 45, 'raisin': 25, 'kiwi': 34}
```

Exercice:

Créer une fonction consolidation qui reçoi en argument deux dictionnaires dict1,dict2 est mise à jour a partie de dict1 à partir de dict2

Structure en python (dictionnaire)



Exercice:

Ecrire un programme en Python qui demande à l'utilisateur de saisir un mot et de lui renvoyer un dictionnaire dont les clés sont les lettres du mot saisi et les valeurs sont le nombre d'apparitions de lettre dans le mot:

Exemple:

anticonstitutionnellement

```
# mot=input("donnez un mot: ")
mot='anticonstitutionnellement'
maList=list(mot)
print(maList)
monDict=dict({})
for lettre in maList:
    if lettre in monDict.keys():
        monDict[lettre]+=1
    else:
        monDict[lettre]=1
print(monDict)
```

Structure en python (dictionnaire)

Exercice 2:

On dispose de deux
fruits=["orange","pomme","fraise","banane"]
stock=[120,130,60,40]
Ecrire un programme qui fait la consolidation des deux listes dans un dictionnaire qui aura comme clés les fruits et pour valeurs les valeurs du stock sous forme:
{ 'orange':120,'pomme' :130,'fraise' :60,'banane' :40}

```
Solution:
 fruits=["orange","pomme","fraise","banane"]
stock=[120,130,60,40]

monDict={}    #création d'un dictionnaire vide
n=len(fruits) #longueur de la liste fruits
for i in range(n):
    monDict[fruits[i]]=stock[i]

print(monDict)
```

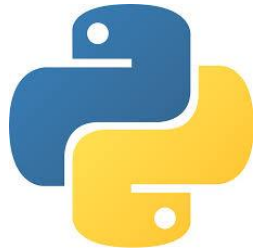
python (tuple)

définition:

Les tuples sont utilisés pour stocker plusieurs éléments en une seule variable.
Tuple est l'un des 4 types de données intégrés dans Python utilisés pour stocker des collections de données, les 3 autres sont Liste, Ensemble et Dictionnaire, tous avec des qualités et une utilisation différentes.
Un tuple est une collection qui est ordered et immuable on peut pas changer les éléménts d'un tuple.
Les tuples sont écrits avec des **parenthéses** .

```
mytuple = ("orange", "pomme", "fraise")
```

python (tuple)



Elément d'un tuple:

Tuple Items
Les éléments Tuple sont ordered onrdonnés, immuables et permettent des valeurs en double.
Les éléments Tuple sont indexés, le premier élément a index [0], le deuxième élément a index [1] etc.

```
mytuple = ("orange","pomme" )
mytuple2=("orange", )
```

```
mytuple=('orange','pomme','fraise')
print(type(mytuple))#<class 'tuple'>
print(len(mytuple)) #3
#balayer les éléments de tuple
for fruit in mytuple:
    print(fruit) print('----------------------')
#recuperer un élément par indice
print(mytuple[0]) #orange
```

python (tuple)

Tranche sur tuple:

```
mytuple=('orange','pomme','fraise','banane','kiwi')
tuple2=mytuple[1:4]
print(tuple2) #('pomme', 'fraise', 'banane')
print(type(tuple2)) #<class 'tuple'>
```

Vérifiez si un élément existe dans un tuple :

Pour déterminer si un élément spécifié est présent dans un tuple, utilisez le mot clé **in**:

```
fruits = (“pomme", "banane", "fraise")

if "pomme" in fruits:
    print(“oui, ‘pomme’ se trouve dans le tuple  fruits")

#set est de type <class 'set'>
```

python (set)

Elément d'un set:

Un set est une collection qui n'est pas ordonnés et non indexée.
Les éléments de sets sont entre **accolades brackets** .

```
fruits = {"apple", "banana", "cherry"}
print(fruits)
```

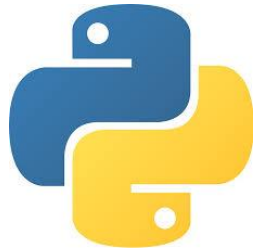
Les ensembles set ne sont pas ordonnés , de sorte que vous ne pouvez pas être sûr dans quel ordre les éléments apparaîtront.

Une fois qu'un ensemble set est créé, vous ne pouvez pas modifier ses éléments, mais vous pouvez ajouter de nouveaux éléments.

```
fruits = {"pomme","banane","fraise","pomme"}
Les éléments dupliqués seront ignoré
```

```
La longueur d'un set est le nombre de ses éléments non dupliqués
Len(fruits) #donne 3
```

python (set)



Balayer les elements d'un set:

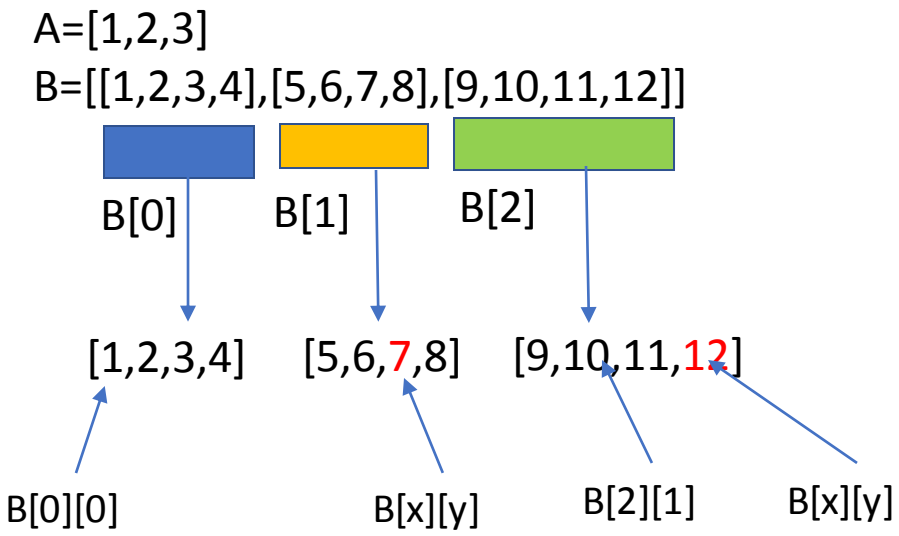
```
for elm in fruits:
    print(elm)
```

```
fruits = {"pomme","banane","fraise","pomme"}
fruits.add("orange")
fruits.remove('pomme')
```

liste imbriquée(nested list) (Matrice)

Matrice:

Une liste peut contenir n'importe quel objet, même une autre liste (sousliste), qui à son tour peut contenir des sous-listes elles-mêmes, etc. Il s'agit de la liste imbriquée.
Une matrice peut être représenté par une liste imbriquée



B	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

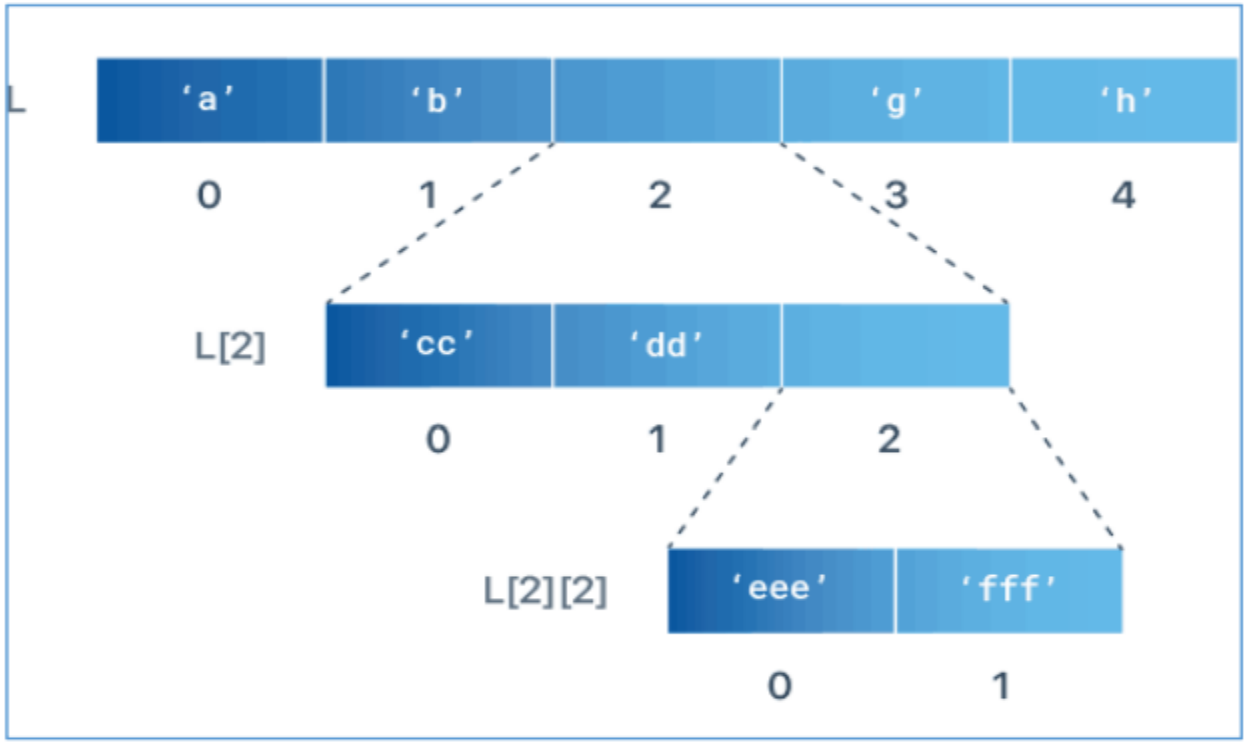
B[1][2]

Len(B) #3 c'est le nombre des lignes
Len(B[0]) #4 c'est le nombre des colonnes

liste imbriquée(nested list)

Exemple :

L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']

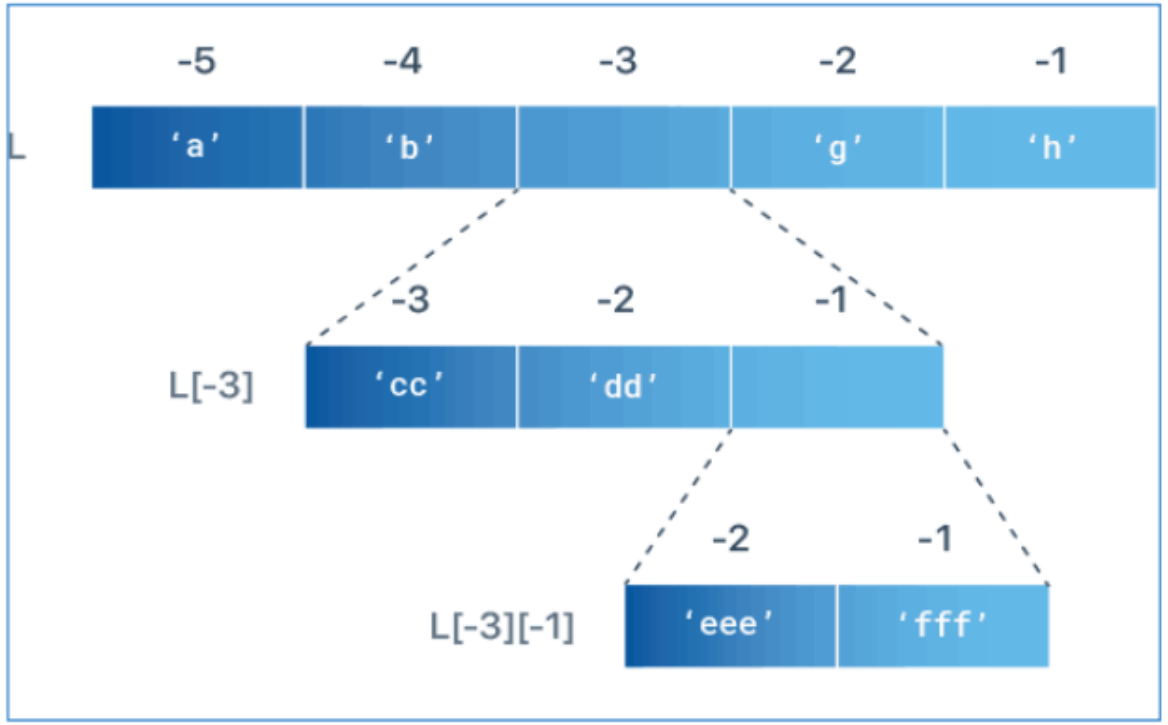



```
L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']
print(L[2]) #['cc', 'dd', ['eee', 'fff']]
print(L[2][2]) #['eee', 'fff']
print(L[2][2][1])#fff
```

liste imbriquée(nested list)

Exemple : avec indices négatifs

L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']




```
L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']
print(L[-3]) #['cc', 'dd', ['eee', 'fff']]
print(L[-3][-1]) #['eee', 'fff']
print(L[-3][-1][-1])#fff
```

liste imbriquée(nested list)

Création et initialisation

Méthode 1:

```
A=[1,2,3]
B=[[1,2,3,4],[5,6,7,8],[9,10,11,12]]
```

Méthode 2:

```
a=[0]*3
print(a) #[0, 0, 0]
b=[a]*4
print(b) #[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Méthode 3:

```
a=[0]*3
for i in range(0,3):
    a[i]=[0]*4
print(a)
```

Méthode 4:

```
a=[]
for i in range(0,3):
    a.append([0]*4)
print(a) #[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

Tableaux in Python (List)

List Comprehension

valeur	collection	condition
List2= [x*2	For x in range(0,10)	If x%2==0]

Exercice 1:

Donner le programme Python qui permet de créer une liste contenant les nombres paires qui sont inferieur à 50

```
#methode classique
print('création par list par methode classique')
list2=[]
for i in range(0,50):
    if i%2==0:
        list2.append(i)
print(list2)
```

```
#list par list comprehension
print(' création par list comprehension ')
list1=[i for i in range(0,50) if i%2==0]
print(list1)
```

Tableaux in Python (List)

Liste imbriquée avec Comprehension

valeur	collection	condition
List2= [x*2	For x in range(0,10)	If x%2==0]

0	0	0	0
0	0	0	0
0	0	0	0

1	2	3	4
5	6	7	8
9	10	11	12

Exercice 1:
Donner le programme Python qui permet de créer une Matrice comme si dessus

```
#methode classique
```

Tableaux in Python (List)

Liste imbriquée avec Comprehension

Exercice 1:

Créer une matrice contenant des zéros de 4 lignes et trois colonnes
utiliser expression Compréhension.

0	0	0	0
0	0	0	0
0	0	0	0

```
b=[[0 for i in range(1,4)]for j in range(4)]
print(b)
```

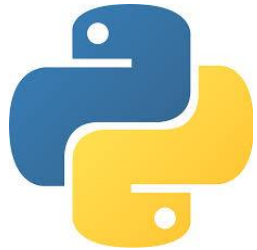
Exercice 2:

Créer une matrice comme ci-dessus utiliser expression Compréhension.

1	2	3	4
5	6	7	8
9	10	11	12

```
b=[[i+j*3 for i in range(1,4)]for j in range(4)]
print(b)
```

Tableaux in Python (List)

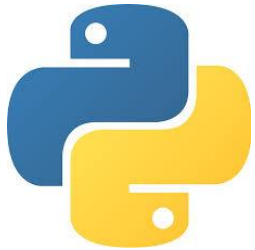


Parcourir les éléments d'une liste imbriquée

```
L = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]
for row in L:
    for col in row:
        print(col, end=' ')
```

```
planets = [['Mercury', 'Venus', 'Earth'], ['Mars', 'Jupiter
', 'Saturn'],
['Uranus', 'Neptune', 'Pluto']]
flatten_planets = []
for sublist in planets:
    for planet in sublist:
        if len(planet) < 6:
            flatten_planets.append(planet)
print(flatten_planets) #['Venus', 'Earth', 'Mars', 'Pluto']
```

Portée des variables in python



Une variable n'est disponible qu'à partir de l'intérieur de la région où elle est créée. C'est ce qu'on appelle la portée.

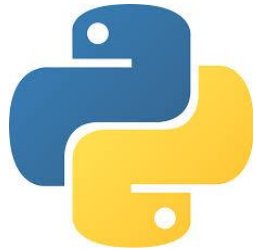
Portée locale (local scope)

Une variable créée à l'intérieur d'une fonction appartient à la portée locale de cette fonction, et ne peut être utilisée qu'à l'intérieur de cette fonction.

```
def myFunction():
    val1=10
    print(val1)
myFunction()
print(val1)
```

```
10
NameError: name 'val1' is not defined
```

Portée des variables in python



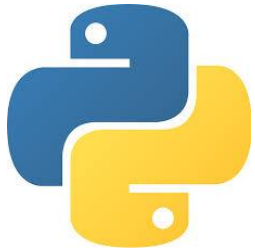
Portée locale (local scope)

```
def myFunction():
    val1=10
    print(val1)
myFunction()
print(val1) #cause erreur
```

Comme expliqué dans l'exemple ci-dessus, la variable val1 n'est pas disponible en dehors de la fonction, mais elle est disponible pour n'importe quelle fonction à l'intérieur de la fonction :

```
def myFunction():
    val1=10
    def fn2():
        print(val1)
    fn2()
myFunction()
```

Portée des variables in python



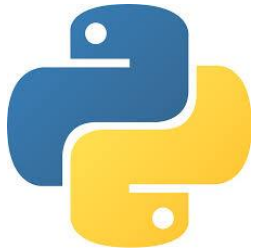
(global keyword) mot clé global

Si vous avez besoin de créer une variable globale, mais que vous êtes coincé dans la portée locale, vous pouvez utiliser le mot clé global.
Le mot clé global rend la variable globale.

```
val1=0
def fn1():
    val1=10
fn1()
print(val1) #val1 contient 0
```

```
val1=0
def fn1():
    global val1
    val1=10
fn1()
print(val1)#val1 contient 10
```

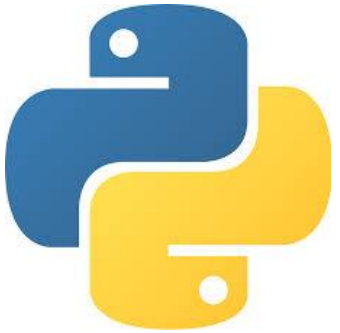
Module in python



C’est quoi Module in python?

un module est identique à une bibliothèque de code.
Un fichier contenant un ensemble de fonctions que vous souhaitez inclure dans votre application.

python manipulation des fichiers



Read file:

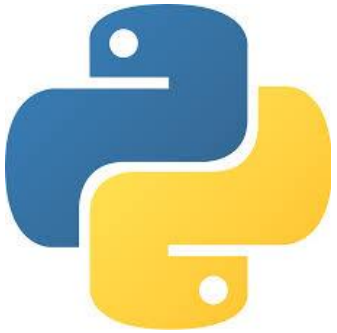
Pour ouvrir le fichier, utilisez la fonction **open()** intégrée() built-in. La fonction **open()** renvoie un objet de fichier, qui a une méthode **read ()** pour lire le contenu du fichier

```
f=open("myText.txt",mode='r’)
print(f.name)
print(f.mode)
myStr=f.read()
print(myStr)
f.close()
```

```
#utilisation de context manager
with open("myText.txt",'r') as f2:
    myStr=f2.read()
print(f2.closed)
```

La propriété f.name contient le nom du fichier
La propriété f.mode contient le mode d’ouverture du fichier
La méthode read() retourne la totalité du contenu du fichier
Après le traitement sur le fichier on doit impérativement fermer le fichier par close()

python manipulation des fichiers



Read avec nbCaractéres méthodes seek,tell,

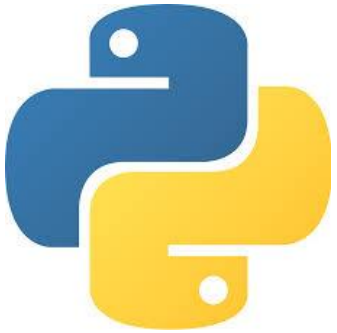
On peut spécifier le nombres de caractères a lire dans la méthode read()

```
f=open("myText.txt",mode='r')
myStr=f.read(10)
print(myStr)#cours pyth
pos=f.tell()
print(pos)#affiche 10
f.close()
```

```
#balayer le fichier
mystr=f.read(10)
while len(mystr):
    print(mystr,"*",sep="")
    mystr=f.read(10)
f.close()
```

```
#utilisation de la methode seek
f=open("myText.txt",mode='r')
f.seek(1)
myStr=f.read(10)
print(myStr)
pos=f.tell()
print(pos)#affiche 11
```

python manipulation des fichiers



readline file:

readline() :permet de lire ligne par ligne

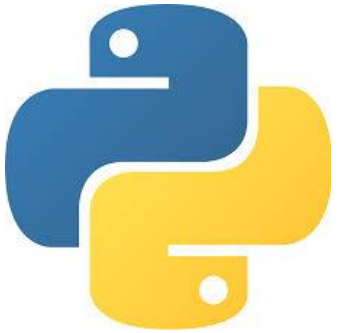
```
f=open("myText.txt",mode='r’)
print(f.name)
print(f.mode)
myStr=f.read()
print(myStr)
f.close()
```

```
#utilisation de context manager
with open("myText.txt",'r') as f2:
    myStr=f2.read()
print(f2.closed) #retourne True
```

```
#pour bilayer les lignes du fichier
f=open("myText.txt",mode='r')
for line in f:
    print(line)
```

La propriété f.name contient le nom du fichier
La propriété f.mode contient le mode d’ouverture du fichier
La méthode read() retourne la totalité du contenu du fichier
Après le traitement sur le fichier on doit impérativement fermer le fichier par close()

python manipulation des fichiers



readlines file:

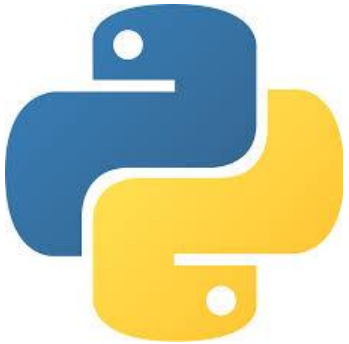
readlines(): retourne une liste qui a pour éléments les lignes du fichier

```
f=open("myText.txt",mode='r')
myLines=f.readlines()
print(myLines)
f.close()
```

```
['cours python\n', 'utiliser la methode open pour\n', 'ouvrire un fichier\n', 'modifier un fichier\n']
```

['cours python\n', 'utiliser la methode open pour\n', 'ouvrire un fichier\n', 'modifier un fichier\n']

python Read and write file



Paramétre mode dans la méthode open() :

"r" - Read - Default value. Ouvrir le fichier en mode lecture erreur si le fihier n'existe pas

"a" - Append – ouvrir le fichier en mode ajout créer le fichier si il n'existe pas

"w" - Write – ouvre le fichier en mode ecriture,céer le fichier si il n'existe pas

"x" - Create - Creates the specified file créer la ficher retourne erreur si le fichier existe

python Read and write file

fichier binaire ou texte

On peut specifier si le fichier doit être traité comme binaire ou texte

"t" - Text - Default value. Text mode
"b" - Binary - Binary mode (e.g. images)

```
f = open("demofile.txt")
# Est equivalent à
f = open("demofile.txt", "rt")
```

. Étant donné que « r » pour lire, et « t » pour le texte sont les valeurs par défaut, vous n'avez pas besoin de les spécifier

python Read and write in file

écrire sur un fichier

Pour écrire à un fichier existant, vous devez ajouter le paramètre mode à la fonction open () :
a - Append – ajouter à la fin du fichier
w - Ecrire - écrasera tout contenu existant du fichier

```
f = open(“myfile.txt", "a")
f.write(“maintenant un contenu est ajouté a la fin du fichier!")
f.close()
#ouvrir et lire le fichier :
f = open(“myfile.txt", "r")
print(f.read())
```

python Read and write in file

écrire sur un fichier

Pour écrire à un fichier existant, vous devez ajouter le paramètre mode à la fonction open () :
a - Append – ajouter à la fin du fichier
w - Ecrire - écrasera tout contenu existant du fichier

```
f = open("demofile3.txt", "w")
f.write("Ahh j ai supprimé le contenu!")
f.close()
#open and read the file after the appending:
f = open("demofile3.txt", "r")
print(f.read())
```

python Read and write in file

créer un nouveau fichier

To create a new file in Python, use the open() method, with one of the following parameters:
Pour créer un nouveau fichier en Python, utilizer la méthode open(), en specifiant le parameter mode par:

x - Create - permet de créer un fichier et retoune une erreur si le fichier existe déja
a - Append - will create a file if the specified file does not exist

w - Write - will create a file if the specified file does not exist

```
f = open("myfile.txt", "x") #le fichier myfile.txt sera créé
```

```
f = open("myfile.txt", "w") #crée le fichier myfile.txt
```

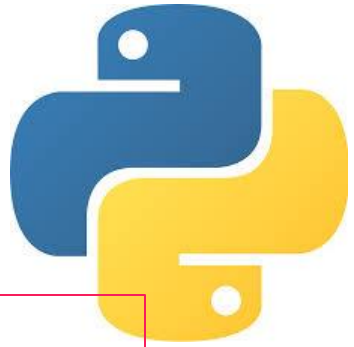
python Read and write in file

copier un fichier binaire cas Image

```
with open("fraise.jpg",'rb') as f:
    with open("fraise2.jpg",'wb') as f2:
        mySize=4096
        line=f.read(mySize)
        while len(line)>0:
            f2.write(line)
            line=f.read(mySize)
print("image est créé.......")
```

```
with open("fraise.jpg",'rb') as f:
    with open("fraise2.jpg",'wb') as f2:

        myBinaireImage=f.read()
        f2.write(myBinaireImage)
print("fraise2.jpg est créée....")
```

python Read and write in file



lire un fichier csv

```
f=open("notesStagiaires.csv",mode='r')
notes=[]
line=f.readline()
while(len(line)>0):
    mylist=line.split(';')
    print(mylist)
    myDict={}
    myDict['code']=mylist[0]
    myDict['nom']=mylist[1]
    myDict['prenom']=mylist[2]
    myDict['noteMath']=mylist[3]
    myDict['notePhysique']=mylist[3][:-1]
    notes.append(myDict)
    line=f.readline()
f.close()
print(notes)
```

	A	B	C	D	E
1	code	nom	prenom	noteMath	notePhysique
2	100	FAHMI	AHMED	12	14
3	101	RAMI	ALI	10	15
4	102	ZAHIR	KAMAL	13	17
5	103	MOURABIT	OMAR	17	15
6	104	ELFADILI	KHALID	8	12
7	105	RAHMANI	IBRAHIM	14	16
8	106	FAOUZI	FATIHA	13	15
9	107	KAMALI	KHADIJA	8	11
10	108	IBRAHIMI	FARID	12	11
11	109	CHAOKI	AHMED	15	17
12	110	CHAHID	SAMI	14	11