Fonctions en python

Exercice 1 : Comparer deux nombres

Écrire la fonction compare() qui reçoit deux valeurs en argument a et b et affiche un message adapté selon que a est supérieur, inférieur ou égal à à b. Faire des essais en utilisant des entiers, des flottants et des chaînes de caractères.

In [1]:

```
def compare(a, b):
        if a < b:
                print(str(a)+" est inférieur à "+str(b))
        elif a > b:
                print(str(a)+" est supérieur à "+str(b))
        else:
                print(str(a)+ " est égale à "+str(b))
```

In [2]:

```
# ou
def compare(a, b):
    res = "égale"
    if a < b :
        res = "inférieur"
    elif a > b :
        res = "supérieur"
    print(a, "est", res, "à", b)
```

In [3]:

```
compare(5,6)
compare(2+3,5)
compare(7,4)
```

```
5 est inférieur à 6
5 est égale à 5
7 est supérieur à 4
```

Exercice 2 : Test de parité

Question 2.1 :

Écrire la fonction estPair() qui affiche si un nombre reçu en paramètre est pair ou non. Faire des essais en utilisant des entiers, des flottants et des chaînes de caractères. Prévoir les résultats.

In [4]:

```
def estPair(a):
        if int(a)%2 == 0:
                print(str(a)+" est pair")
        else:
                print(str(a) + " est impair")
```

In [5]:

```
# ou
def estPair(a):
    res = "impair"
    if int(a)%2 == 0:
        res = "pair"
    print(a, "est", res)
```

In [6]:

```
estPair(4)
estPair(5)
```

```
4 est pair
5 est impair
```

Question 2.2 :

Ré-écrire la fonction estPair() pour que cette dernière renvoie True si le nombre reçu en paramètre est pair, False sinon. Faire des essais en utilisant des entiers, des flottants et avec des chaînes de caractères. Prévoir les résultats.

In [7]:

```
def estPair(a):
        if a%2 == 0:
                return True
        else:
                return False
```

Essayer d'avoir toujours qu'un seul retour dans une fonction

In [8]:

```
def estPair(a):
    res = False
    if a%2 == 0:
        res = True
    return res
```

In [9]:

```
# ou
def estPair(a):
    return (a%2 == 0)
```

In [10]:

```
estPair(3)
```

Out[10]:

```
False
```

In [11]:

```
estPair(2)
```

Out[11]:

```
True
```

In [12]:

```
estPair(2.5)
```

Out[12]:

```
False
```

Exercice 3 : Moyenne de deux nombres

Écrire la fonction qui reçoit deux nombres en arguments et qui calcule et renvoie leur moyenne. Quel est le problème si les deux nombres sont entiers? Quelle solution proposez-vous?

In [13]:

```
def moyenne(a, b):
    moyenne = (a+b)/2
    return moyenne
```

In [14]:

```
print(moyenne(3,4))
print(moyenne(2,4))
```

```
3.5
3.0
```

Même si le résultat est un entier, la fonction renvoie un float.

In [15]:

```
def moyenne(a, b):
    moyenne = (a+b)/2
    if (a+b)%2 == 0:
        moyenne = int((a+b)/2)
    return moyenne
```

In [16]:

```
print(moyenne(3,4))
print(moyenne(2,4))
```

```
3.5
3
```

Exercice 4 : Année bissextile

Écrire une fonction qui permet de déterminer si une année est bissextile.

On rappelle qu'une année est bissextile si elle est divisible par 4 mais n'est pas divisible par 100 sauf si elle est divisible par 400 Ainsi 2008 était bissextile, 1900 n'était pas bissextile et 2000 était bissextile.

In [17]:

```
def bissextile(année):
        if (int(année)%4 == 0 and int(année)%100 != 0) or int(
année)%400 == 0:
                print(str(année)+" est une année bissextile")
        else:
                print(str(année) + " n'est pas une année bisse
xtile")
```

In [18]:

```
bissextile(1900)
bissextile(2000)
```

```
1900 n'est pas une année bissextile
2000 est une année bissextile
```

Exercice 5 : Lancer de dés

Le but de cet exercice est de créer des fonctions permettant de simuler des lancers de dés. Lancer un dé correspond, d'un point de vue informatique, à tirer un entier aléatoire entre 1 et 6 inclus. Ceci se fait à l'aide de la fonction randint() contenue dans la bibliothèque random. Le code suivant permet de simuler le lancer d'un dé et d'afficher le résultat du lancer.

```
#nécessaire pour pouvoir utiliser la fonction randint()
from random import *
de = randint(1,6)
print('lancer : ' + str(de))
```

Question 5.1 :

Créer une fonction qui simule le lancer de deux dés et renvoie la somme. Utiliser cette fonction pour afficher le résultat d'un lancer de deux dés.

In [19]:

```
from random import *
```

In [20]:

```
def sommeDes():
        de1 = randint(1,6)
        de2 = randint(1,6)
        return de1 + de2
```

In [21]:

```
print(sommeDes())
```

```
4
```

Question 5.2 :

Créer une fonction qui simule le lancer d'un nombre quelconque de dés donné en paramètre. Utiliser cette fonction pour afficher le résultat d'un lancer de deux dés puis de trois dés.

In [22]:

```
from random import *
```

In [23]:

```
def lanceDes(nb):
        for i in range(nb):
                print(randint(1,6))
```

In [24]:

```
print('lancer de 2 dés')
lanceDes(2)
print('lancer de 3 dés')
lanceDes(3)
```

```
lancer de 2 dés
5
4
lancer de 3 dés
5
6
2
```

Exercice 6 : Suite de carrés

Écrire une fonction qui permet d'afficher la suite des carrés jusqu'à n^2 où n est un entier choisi par l'utilisateur.

L'affichage se fera sous la forme 0 – 1 – 4 – 9 – 16 – 25 – 36 – 49 – 64 – 81 – 100...

Dans l'exemple suivant l'entier n est égal à 6 : 0–1–4–9–16–25–36.

In [25]:

```
def carres(n):
        affich = "0"
        for i in range(1, n+1):
                affich += " - "+str(i*i)
        print(affich)
```

In [26]:

```
carres(10)
carres(6)
```

```
0 - 1 - 4 - 9 - 16 - 25 - 36 - 49 - 64 - 81 - 100
0 - 1 - 4 - 9 - 16 - 25 - 36
```

Exercice 7 : Produit d'entiers

Écrire une fonction produit() qui calcule et renvoie le produit n1∗(n1+1)∗...∗n2 (tels que 1<=n1<= n2) des entiers compris entre n1 et n2 inclus.

In [27]:

```
def produit(n1, n2):
    p = 1
    if(not(1 <= n1 <= n2)):
        print("Erreur valeurs d'entrée")
        p = 0
    else:
        for i in range(n1,n2+1):
            p *= i

    return p
```

In [28]:

```
print(produit(10, 20))
```

```
6704425728000
```

Exercice 8 : Échange de variables

Écrire une fonction swap(u,v) qui sert à échanger les valeurs de 2 variables u et v passées en arguments.

In [29]:

```
def swap(a, b):
    c = a
    a = b
    b = c
```

In [30]:

```
# ou
def swap(a,b):
    a,b = b,a
```

In [31]:

```
a = "Hello"
b = 45
print(a,b)
swap(a,b)
print(a,b)
```

```
Hello 45
Hello 45
```

Cette fonction est-elle utile ?

Elle ne sert à rien car elle ne va pas pouvoir modifier les valeurs des variables globales

In [32]:

```
def swap(a,b):
    return b,a
```

Ici, on affiche les variables dans un autre ordre.

In [33]:

```
a = "Hello"
b = 45
print(a,b)
print(swap(a,b))
print(a,b)
```

```
Hello 45
(45, 'Hello')
Hello 45
```

À quels types s'applique-t-elle ?

A tous les types.

Exercice 9 : Comptage des éléments d'un tableau

Écrire une fonction nbPairImpair() qui renvoie le nombre d'élément(s) pair(s) et le nombre d'élément(s) impair(s) dans le tableau reçu en argument.

In [34]:

```
def nbPairImpair(tab):
        pair = 0
        impair = 0
        for nb in tab:
                if nb % 2 == 0:
                        pair += 1
                else:
                        impair += 1
        return pair, impair
```

In [35]:

```
# ou
def nbPairImpair(tab):
        pair = 0
        for nb in tab:
                if nb % 2 == 0:
                        pair += 1
        return pair, len(tab)-pair
```

In [36]:

```
tab =  [12, 21, 10, 11, 0, 1, 6, 8]
tup = nbPairImpair(tab)
print("il y a" ,tup[0], "pairs et", tup[1], "impairs")
```

```
il y a 5 pairs et 3 impairs
```

Exercice 10 : Décalage des éléments d'un tableau à droite

Écrire une fonction decaleCircDroite() qui réalise le décalage circulaire vers la droite d'un tableau d'entiers. Voici un exemple d'utilisation de cette fonction :

- Avant décalage circulaire a droite [12, 21, 10, 11, 0, 1, 6, 8]
- Après décalage circulaire a droite [8, 12, 21, 10, 11, 0, 1, 6]

In [37]:

```
def decaleCircDroite(tab):
    for i in range(-1, - len(tab), -1):
        tab[i], tab[i + 1]= tab[i + 1], tab[i]
```

In [38]:

```
tab = [12, 21, 10, 11, 0, 1, 6, 8]
print(tab)
decaleCircDroite(tab)
print(tab)
```

```
[12, 21, 10, 11, 0, 1, 6, 8]
[8, 12, 21, 10, 11, 0, 1, 6]
```

Exercice 11 : Rappel sur binaire

Question 11.1 : Du binaire vers le décimal.

Écrire une fonction bin2Dec() qui permet de convertir une chaîne de caractères contenant la représentation binaire d'un nombre (codage entier naturel) en sa représentation décimale.

Exemple d'utilisation :

```
nBin='10000001'
nDec = bin2Dec(nBin)
print('Le nombre binaire',nBin, 'se convertit en base 1
0 :', nDec)
```

In [39]:

```
def bin2Dec(nBin):
        nDec = 0
        for i in range(len(nBin)):
                if nBin[i] == "1":
                        nDec += 2**i
        return nDec
```

In [40]:

```
nBin = "10000001"
nDec = bin2Dec(nBin)
print('Le nombre binaire',nBin, 'se convertit en base 10 :', n
Dec, "Résultat juste ? ", (nDec == 129))
```

```
Le nombre binaire 10000001 se convertit en base 10
: 129 Résultat juste ?  True
```

Question 12.2 : Du décimal vers le binaire.

Écrire une fonction qui calcule l'écriture en base 2 d'un nombre entier positif passé en argument sous sa forme décimale. Le résultat pour 5 sera 101.

In [41]:

```
def dec2Bin(nb):
    binaire = ""
    while nb > 1:
        binaire = str(nb % 2) + binaire
        nb = nb // 2
    return "1" + binaire
```

In [42]:

```
nDec = 10
nBin = dec2Bin(nDec)
print('Le nombre decimal',nDec, 'se convertit en base 2 :', nB
in, "Résultat juste ? ", (nBin == "1010"))
```

```
Le nombre decimal 10 se convertit en base 2 : 1010
Résultat juste ?  True
```

In [ ]:

```

```

In [ ]:

```

```