Python, 2nde, Exercices, Fiche 1

Rappels calculs élémentaires en Python

```
>>> 4*5 # multiplication
20
>>> 8/2 # division décimale
4.0
>>> 9//2 # quotient de la division euclidienne
4
>>> 9%2 # reste de la division euclidienne
1
>>> 5**2 # puissance
25
```

Exercice 1

On considère les fonctions python suivantes :

```
def fonction1(nombre):
    return nombre*3

def fonction2(nombre):
    return nombre**2

def fonction3(nombre):
    return nombre/2

def fonction4(nombre):
    return nombre//2

def fonction5(nombre):
    return nombre%2
```

Que renvoient les commandes suivantes ? Vérifier vos réponses en exécutant les commandes après avoir enregistrer les fonctions.

```
>>> fonction1(7)
...
>>> fonction2(7)
...
>>> fonction3(7)
...
>>> fonction4(7)
...
>>> fonction5(7)
...
```

Les boucles for ... in range .....

On considère les fonctions suivantes :

```
def boucle1():
    for i in range(5):
        print(i)

def boucle2():
    for i in range(3):
        print(i)

def boucle3():
    for i in range(1,3):
        print(i)

def boucle4():
    for i in range(2,8):
        print(i)

def boucle5():
    for i in range(3,10,2):
        print(i)

def boucle6():
    for i in range(2,7,3):
        print(i)
```

Taper et exécuter vos fonctions. Tester les.

Ecrire les fonctions boucle7(), boucle8() et boucle9() de façon à avoir les résultats suivants après leur exécution.

```
>>> boucle7()
0
1
```

```
2
3
4
5
6
>>> boucle8()
10
11
12
13
14
>>> boucle9()
20
23
26
```

Les tests : If ..... then .....else

On considère les fonctions suivantes :

```
def test1(x):
    if x==4:
        return True
    else :
        return False

def test2(x):
    if x==8:
        return True
    else :
        return False

def test3(a):
    b=7
    if a==b:
        return True
    else :
        return False

def test4(x):
    if x!=8:
        return True
    else :
        return False

def test5(x):
    if x*2==8:
        return True
    else :
```

```
        return False
```

que vont retourner les commandes suivantes ?

```
>>> test1(4)
...
>>> test1(8)
...
>>> test2(4)
...
>>> test2(8)
...
>>> test3(2)
...
>>> test3("b")
...
>>> test3(7)
...
>>> test4(8)
...
>>> test4(9)
...
>>> test5(8)
...
>>> test5(4)
...
```

Ecrire une fonction pair()) et une fonction impair() qui permettent de déterminer si un nombre est pair ou impair.

Exemple d'exécution :

```
>>> pair(8)
True
>>> pair(5)
False
>>> impair(8)
False
>>> impair(5)
True
```

Ecrire une fonction multiple_de_3() qui permet de déterminer si un nombre est un multiple de 3:

```
>>> multiple_de_3(15)
True
>>> multiple_de_3(10)
False
```