Exercice 1 : Qu'obtient-on en exécutant ces codes ?

Code	***Shell après exécution***
`print(5!= 2+2)`	

`n = 8`
`condition = (n%2 == 0 and n%4 == 0)`
`print(condition)`	

`mot = "Bienvenue"`
`condition = (mot == "Welcome")`
`print(condition)`	

`while 5 > 4 :` `    print("Vive les vacances")`	

`n = 0`
`while 3 != 1+1 :`
`    n = n + 1`
`    print(n)`	

`condition = (8 > 7)`
`while condition :`
`    print(condition)`	

`while False :` `    print("Tu as gagné 100 000 €")`	

`n = 100`
`while n > 1 :`
`    n = n // 2`
`    print(n)`	

```
somme = 1000
annee = 2020
while somme < 2000 :
    annee = annee + 1
    somme = somme + somme * 15 / 100
print(f"Tu auras {somme:.2f} € en {annee}")
```

Exercice 2 :

Ecrire un code utilisant une structure while qui permet d'obtenir le même résultat que le script ci-contre :

```
for i in range(100) :
    print(i)
```

Exercice 3:

Réécrire le code de la fonction *additionne()* ci-contre, en utilisant une structure while.

```
def additionne(n) :
    somme = 0
    for i in range(n) :
        somme = somme + i
    return somme

s = additionne(4)
print(s)
```

Exercice 4 :

1 **Décrire comme dans l'exemple 1 ci-dessus ce qui se passe lors de l'exécution à la main du script.**

```
a = 1
while a < 11:
    a = a + 4
    print(a)
```

a **prend la valeur** .....

2 Quel affichage obtient-on après exécution de chacun des scripts suivants ?

Script 1

```
S = 1
while S < 10:
    S = S * 2
print(S)
```

Script 2

```
S = 1
while S < 10:
    S = S *2
    print(S)
```

3 Avant exécution du script suivant, n a pour valeur 0 et S a pour valeur 1.

```
while P < 100:
    n = n + 1
    P = P * 5
```

a) On exécute le script à la main.
Compléter au fur et à mesure le tableau suivant :

n	0			
P	1			
P < 100				

b) On dit que la variable *n* est un compteur. Pourquoi ?

4 Le script ci-dessous définit une fonction d'un argument n entier naturel.

```
def mystere(n):
    while n >= 11:
        n = n - 11
    return n
```

a) Que renvoient :

```
>>> mystere(20)
>>> mystère(48)
>>> mystere(11)
>>> mystere(7)
```

b) Expliquer ce que renvoie cette fonction.

5 On définit la fonction Python puiss.

```
def puiss(n):
    a = 1
    while a <= n:
        a = a * 2
    return a
```

a) Que renvoient :

```
puiss(10)
puiss(25)
puiss(1000)
```

b) Que renvoie cette fonction (entourer la bonne réponse) ?

(1) la plus grande puissance de 2 inférieure ou égale à *n* ;

(2) la plus petite puissance de 2 inférieure ou égale à *n* ;

(3) la plus grande puissance de 2 supérieure à *n* ;

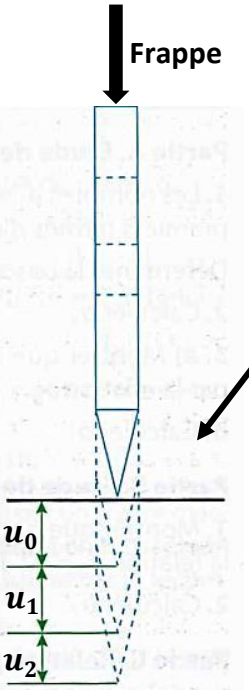
(4) la plus petite puissance de 2 supérieure à *n*.

Exercice 5:

Une ruche contient 5000 abeilles au 1er juillet 2021. Durant tout l'été, le nombre d'abeilles diminue de 1 % chaque jour. Ecrire un code qui permette de calculer le nombre de jours à partir duquel les abeilles seront moins de 3000.

Exercice 6:

Un vigneron décide d'installer ses piquets à l'aide d'un enfonce pieu pendulaire à mat éleveur branché sur la prise de force du tracteur.



Le pieu est introduit initialement dans un trou de 10 cm de profondeur. A chaque frappe, il s'enfonce un peu plus.

$u_0 = 10$,

$u_1 =$ **déplacement** vertical du pieu après la 1ère frappe, $u_2 =$ **déplacement** vertical du pieu après la 2nde frappe (voir figure ci-contre),

…….

$u_n =$ déplacement vertical du pieu après la nième frappe.

Le sol étant de plus en plus dense, lors de l'enfoncement, entre 2 frappes successives, le déplacement vertical diminue à chaque fois de 15 %.

Ainsi u_1 est 15 % inférieur à u_0 , u_2 est 15 % inférieur à u_1, etc …. u_{n+1} est 15 % inférieur à u_n

L'enfoncement total du pieu après n frappe est : $E_n = u_0 + u_1 + u_2 + \cdots .. + u_n$

⇨ Ecrire un code qui permette de calculer le nombre n de frappes pour que l'enfoncement total soit de 30 cm.

Exercice 7 :

Compléter le tableau en indiquant l'état de la variable a et du test indiqué (T pour True et F pour False).

1 ★☆☆

```
a = 1
while a < 12:
    a = 5*a
```

a	1	.....	.....
a<12	T	.....	.....

2 ★☆☆

```
a = 32
while a > 11:
    a = a - 11
```

a	.....	.....	.....
a>11	.....	.....	.....

3 ★☆☆

```
n = 60
while n%2 == 0:
    n = n//2
```

a	.....	.....	.....
n%2==0	.....	.....	.....

Compléter en indiquant l'affichage obtenu après exécution des instructions.

4 ★☆☆

```
a = 1
while a < 12:
    a = 6*a
    print(a)
```

.....

.....

5 ★☆☆

```
a = 43
while a > 13:
    a = a - 13
print(a)
```

.....

.....

6 ★☆☆

```
n = 120
while n%2 == 0:
    n = n//2
print(n)
```

.....

.....