Travaux Pratiques 4 : boucle while

Exercice 1:

Deviner l'affichage du programme suivant avant de l'exécuter :

```
x=0
while x<=5:
    x=x+1
    print(x)
```

Exercice 2:

On s'intéresse au jeu suivant : la machine choisit un entier au hasard entre 1 et 100 et le joueur doit le deviner.

1. Compléter le programme suivant afin de pouvoir y jouer !

```
import numpy.random as rd   # bibliothèque nécessaire pour simuler du hasard
cible= rd.randint(1,101)   # l'ordinateur choisit un entier au hasard entre 1 et 100
                              et le stocke dans la variable cible
essai=int(input('Entrez votre proposition'))
while ..............
    if .................. :
        print("Trop grand")
    else:
        print("Trop petit")
    essai=int(input('Entrez votre proposition'))
print("Gagné")
```

2. Ajouter une variable qui compte le nombre d'essais du joueur et l'afficher.
3. Comment optimiser la stratégie du joueur pour faire le moins d'essais possibles ?
   En combien d'essais au maximum peut-on alors y arriver quelque soit la cible ?
4. ***bonus* : Modifier le programme pour que le joueur n'ait droit qu'à 7 essais maximum. On affichera "Perdu" si le joueur n'a toujours pas trouvé au septième essai.

Exercice 3:

On considère la suite définie par $u_0 = 4$ et $\forall n \in \mathbb{N}$, $u_{n+1} = u_n^2 + 1$.

1. Ecrire une fonction d'en-tête `def suite(n):` qui à un entier n renvoie la valeur de u_n.
2. Ecrire une fonction qui prend en entrée un réel x et qui renvoie la valeur du premier terme de la suite qui soit supérieur ou égal à x. *On pourra s'inspirer du 1., mais en utilisant une boucle* `while`.
   Pour vérifier votre fonction : $x = 17$ renvoie 17 et $x = 1000$ renvoie 84101.
3. Modifier votre deuxième fonction pour qu'elle renvoie de plus l'indice du terme de la suite.
   (Pour vérifier : pour x=1000 on trouve n=3). A quelle vitesse diverge cette suite ?

Exercice 4:

Soit la suite u définie par $u_0 = 3/4$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = u_n - u_n^2$. On admet que la suite u est décroissante et converge vers 0. Ecrire un programme qui détermine le premier entier n pour lequel $u_n < 10^{-3}$.
Qu'en déduit-on sur la vitesse de convergence de la suite ?

Exercice 5:

Soit u une suite convergeant vers ℓ telle que pour tout $n \in \mathbb{N}$, $|u_n - \ell| \leq (\frac{2}{5})^n$.

1. Déterminer sur votre feuille, le premier entier n tel que $|u_n - \ell| \leq 10^{-3}$.
   A l'aide de python, calculer le.
2. Ecrire un programme python qui, à l'aide d'une boucle `while`, trouve le même entier.
3. Quel terme de la suite choisir pour obtenir une valeur approchée de ℓ à 10^{-3} près ?

Exercice 6: `eml E 2018`

Soit la suite u définie par $u_0 = 4$ et $\forall n \in \mathbb{N}$, $u_{n+1} = \ln(u_n) + 2$. L'étude mathématique permet de justifier qu'elle est bien définie, qu'elle converge vers un réel $b > 1$, et que pout tout $n \in \mathbb{N}$, $0 \leq u_n - b \leq \frac{1}{2^{n-1}}$.

1. Écrire une fonction d'en-tête `def suite(n):` qui, prenant en argument un entier n de $\mathbb{N}$, renvoie la valeur de u_n.
   Rappel : la fonction ln s'obtient via `np.log()` (penser à importer la bibliothèque numpy `import numpy as np`).
2. Recopier et compléter la ligne 3 de la fonction suivante afin que, prenant en argument un réel `epsilon` strictement positif, elle renvoie une valeur approchée de b à `epsilon` près.

```
1. def valeur_approchee(epsilon):
2.     n = 0
3.     while ........:
4.         n = n+1
5.     return suite(n)
```

Exercice 7:

La syntaxe `rd.randint(a,b+1)` renvoie un entier au hasard entre a et b, dès que la bibliothèque `numpy.random` est chargée via la commande `import numpy.random as rd`

1. Ecrire un programme python qui simule des lancers de 2 dés jusqu'à l'obtention d'un double, et renvoie le nombre de lancers effectués.
2. * *bonus :* Compléter le script précédent afin d'avoir une valeur approchée du nombre moyen d'essais à faire pour obtenir le premier double.

Exercice 8: *Sommes : encore et toujours*

1. Ecrire une fonction python qui à un entier n renvoie la somme $\sum_{k=1}^{n} \frac{1}{k}$.
2. Ecrire alors une fonction python qui à un réel $A \in \mathbb{R}$ renvoie la valeur du premier $n \in \mathbb{N}^*$ tel que la somme $\sum_{k=1}^{n} \frac{1}{k}$ soit strictement supérieure à A. (pour vérifier : pour $A = 1.5$ $n = 3$, pour $A = 10$, $n = 12367$).
3. A quelle vitesse la somme diverge vers $+\infty$?

Exercice 9: *pour s'entraîner*

L'algorithme de Syracuse consiste à itérer l'opération suivante pour un entier naturel n non-nul donné :

$n \mapsto \begin{cases} \frac{n}{2} & \text{si } n \text{ est pair} \\ 3n+1 & \text{si } n \text{ est impair} \end{cases}$ On conjecture que quelque soit l'entier de départ considéré, au bout d'un certain nombre d'itérations, on arrive à l'entier 1.

1. Sur votre feuille, donner les entiers successifs apparaissant dans cet algorithme, lorsque l'entier initial est 3. Combien d'itérations sont nécessaires pour obtenir le premier 1 ?
2. Ecrire alors un programme python qui demande un entier n non-nul à l'utilisateur, qui effectue l'algorithme de Syracuse et qui affiche tous les entiers obtenus jusqu'au premier 1.
3. Compléter le programme précédent afin qu'il donne également le nombre d'itérations effectuées jusqu'à l'obtention du premier 1.

Exercice 10: ** `Edhec E 2006` *petite révision sur for*

Compléter la déclaration de fonction suivante pour qu'elle renvoie la valeur de $\sum_{k=0}^{n-1} \frac{a^k}{k!} e^{-a}$ à l'appel de sigma (a, n).

On suppose que la librairie `numpy` a été chargée. La commande pour la fonction exponentielle est : `np.exp()`

```
def sigma(a,n):
    p=1
    y=1
    for k in range(1,n):
        p= p*a/ k
        y=..........
    y=.........
    return y
```

Exercice 11: *Méthode de dichotomie (du grec ancien "couper en deux")*

```
import numpy as np

def f(x):
    return np.exp(x)-2

# Entrée de l'intervalle de recherche [a,b] et de
# la précision voulue e
e=float(input('Entrer la précision voulue'))
a=float(input('Entrer la valeur de a'))
b=float(input('Entrer la valeur de b'))

while np.abs(b-a)>e:
    c=(a+b)/2
    if f(b)*f(c)<=0:
        a=c
    else:
        b=c

print('Une valeur approchée de la solution à ',
e,'près est ',c)
```

1. Représenter graphiquement la fonction f sur sa feuille.
2. Exécuter à la main le programme pour les valeurs initiales `a`=-2, `b`=2 et `e`=1/2.
   Faire de même avec `a`=-2, `b`=2 et `e`=1/4.
   Que constatez-vous ?
3. Utiliser la représentation graphique de f sur sa feuille pour visualiser toutes les valeurs successives de `a`, `b`, et `c`.
   Que représentent `a`, `b` et `c` au cours du programme ? ?
4. Quelle est la question à laquelle répond ce programme ?