Devoir d'informatique

Exercice 1 – Structure If

Question 1 *Implémenter la fonction* `f_conditionnel(x:int)-> int` *correspondant à la fonction mathématique suivante :*

$$f : x \to \begin{cases} 2 & \text{si } x < 0 \\ 10 & \text{si } x = 0 \\ -x+2 & \text{si } x \in]0;3] \\ -1 & \text{si } x > 3 \end{cases}$$

Correction

```
def f_conditionnel(x):
    if x<0:
        return 2
    elif x == 0:# == et non =
        return 10
    elif x<3:#on sait que x >0 et x!=0, on est donc dans ]0, 3[
        return -x+2
    else : # x>0 et x!=0 et x>= 3
        return -1
```

Exercice 2 – Structure for

Question 2 *Ecrire une fonction* `multiples_3(n:int) ->None` *affichant de tous les multiples de 3 compris entre 0 et n inclus. On utilisera une boucle* `for`.

Correction

```
def multiples_3(n):
    somme = 0
    for i in range(n+1):
        if i%3 == 0 :
            print(i)
```

Exercice 3 – Structure while

Question 3 *Ecrire une fonction* `multiples_3w(n:int) ->None` *affichant de tous les multiples de 3 compris entre 0 et n. On utilisera une boucle* `while`.

Correction

```
def multiples_3w(n):
```

```
    i = 0
    while i <= n:
        if i%3 == 0 :
            print(i)
        i = i+1
```

Exercice 4 – Structure for ou while

Question 4 *Ecrire une fonction* `somme_pairs(n:int)->n` *calculant la somme des entiers pairs de 1 à n exclus.*

Correction

```
def somme_pairs(n):
    somme = 0
    for i in range(n):
        if i%2 == 0 :
            somme = somme + i
    return somme
```

Exercice 5 – Implémentation d'une suite

On pose $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = \frac{1}{2}\left(u_n + \frac{n+1}{u_n}\right)$.

Question 5 *Ecrire une fonction* `u(n:int)->n` *permettant de renvoyer le nième terme de la suite u.*

Correction

```
def u(n):
    res = 1
    for i in range(n):
        res = 0.5*(res+(i+1)/res)
    return res
```

Exercice 6 –

On donne la fonction suivante :

```
def inv(n):
    """Somme des inverses des n premiers
       entiers naturels non nuls"""
    s = 0
    for k in range(n):
        x = 1/k
    s = s+x
    return s
```

Question 6 *Donner l'évolution des variables k et x en fin d'itération lors de l'appel suivant* `inv(4)`*. Quelle est la valeur renvoyée par la fonction ?*

Correction

Au début de la première itération, `k` vaut 0. La division par 0 renvoit une erreur.

Réponse acceptée :

- k vaut 0 et x vaut 1/0 :(
- k vaut 1 et x vaut 1/1 ;
- k vaut 2 et x vaut 1/2 ;
- k vaut 3 et x vaut 1/3.

La fonction retourne 1/3.

Question 7 *Corriger la fonction pour qu'elle réponde à l'objectif anoncé.*

Correction

```
def inv(n):
    """Somme des inverses des n premiers
       entiers naturels non nuls"""
    s = 0
    for k in range(1,n):
        x = 1/k
        s = s+x
    return s
```

Exercice 7 – ADN

Un brin d'ADN est composé de bases nucléiques notées `'a'`, `'c'`, `'g'` ou `'t'`.

Question 8 *Ecrire une fonction* `verifie(adn:str)->bool` *qui renvoie* `True` *si la chaine ne contient que des bases nucléiques* `'a'`, `'c'`, `'g'` *ou* `'t'`. *Sinon, la fonction renvoie* `False`.

Correction

```
def verifie(adn):
    bases = 'acgt'
    for lettre in adn :
        if lettre not in bases :
            return False
    return True
```

Question 9 *Ecrire une fonction* `ajout(adn:str,base:str)->str` *qui ajoute les bases* `base` *à l'adn* `adn` *et qui renvoie la chaîne résultante. Ainsi* `ajout('acgt','gt')` *renvoie 'acgtgt'.*

Correction

```
def ajout(adn,base):
    res = adn
    for lettre in base :
        res = res+lettre
    return res
```

Question 10 *Ecrire une fonction* `compte(adn:str,base:str)->int` *qui renvoie le nombre d'occurences de la base. La fonction renverra* `-1` *si la chaîne n'est pas valide.*

Correction

```
def compte(adn,base):
    if not(verifie(adn)) :
        return -1
    cpt = 0
    for lettre in adn :
        if lettre == base :
            cpt = cpt+1
    return cpt
```

Question 11 *Ecrire une fonction* `proportion(adn:str,base:str)->int` *qui renvoie la proportion de la base* `base` *dans le brin d'adn. La fonction renverra* `0` *si la chaîne n'est pas valide.*

Correction

```
def proportion(adn,base):
    if not(verifie(adn)) :
        return 0
    return compte(adn,base)/len(base)
```