Exercice 1

mécanisme d'exception

Les exercices suivants s'appuient sur :

- Le cours *mise au point d'un programme* : allophysique.com > python > mise au point programme
- Le cours *modularité* : allophysique.com > nsi > langages > modularité
- Les flash cards sur le theme : allophysique.com > python > Flash card 6
- Le projet *nombres premiers et tests unitaires* : allophysique.com > Posts > Python unittest

On donne le script 1 suivant :

```
def divise(a,b):
    return a/b
```

Si on execute l'instruction suivante : `divise(10,0)`, la console affiche :

```
ZeroDivisionError      Traceback (most recent call last)
<ipython-input-30-8044e08140eb> in <module>
----> 1 print(divise(10,0))

<ipython-input-29-8590a608aa42> in divise(a, b)
      1 def divise(a,b):
----> 2     return a/b

ZeroDivisionError: division by zero
```

1.1 Quel est le message d'exception qui est renvoyé par le Traceback ? Quelle ligne du script pose problème ?

On ne possède pas d'information sur la fonction `divise`. En particulier, on ne sait pas dans quel ordre doivent être mis les arguments. On ne sait pas si on peut écrire `divise(10,0)` ou bien `divise(0,10)` pour faire 0/10.

1.2 Quel mécanisme d'exception try-except pourrait-on écrire à la place de l'instruction `division(10,0)` ?

On donne le script 2 suivant :

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
            print('erreur de type ValueError')
    except SyntaxError:
            print('erreur de type SyntaxError')
       else:
            print('erreur d'un autre type)
```

L'utilisateur du programme rentre le caractère 'Q' :

```
Please enter a number: Q
```

1.3 Qu'est ce qui est affiché en console juste après la saisie de Q ?

1.4 Le programme s'interrompt-il ?

Exercice 2

Projet nombres premiers

On dispose d'une fonction `prem1` dont le prototypage est le suivant :

```
"""retourne True si le nombre N passé en argument est premier False sinon
Param:
N : int
        nombre à tester
Return:
True ou False
"""
```

Et d'une fonction `liste_prime1` donnée ici :

```

def liste_prime1(a,b):
    """retourne une liste de nombre premiers compris entre a et b
    """

    L=[]
    for j in range(a,b+1):
        if prem1(j):
            L.append(j)

    return L
```

On souhaite mesurer le temps mis par la fonction `liste_prime1` pour calculer cette liste.

2.1 L'efficacité de cette fonction, `liste_prime1`, dépend-elle de celle de la fonction `prime1` ? Expliquer.

2.2 Où doit-on placer les 3 instructions suivantes pour calculer la valeur du temps mis par cette fonction ?

- instruction 1 : `import time`
- instruction 2 : `t = time.time()`
- instruction 3 : `print(time.time()-t)`

2.3 Expliquer ce que fait l'instruction 3

Exercice 3

L'algorithme d'Euler sur les nombres premiers

Le légendaire mathématicien suisse Léonhard EULER(1707-1783), proposait la formule suivante pour obtenir des nombres premiers : pour tout entier naturel n,

$$f(n) = n^2 - n + 41$$

3.1 Compléter le script de la fonction `euler` pour que celle-ci calcule l'image d'un entier n par cette fonction `f`

```
def euler(n):
  return ...
```

3.2 Compléter le script de la fonction `liste_prime_euler` qui créé une liste de nombres premiers compris entre les bornes a et b à partir de cette formule.

```
def liste_prime_euler(a,b):
    """retourne une liste L de nombres calculés selon
    l'equation d'Euler, compris entre a et b.
    Params:
    a,b: int
        les bornes pour le calcul
    Return:
    L: liste d'entiers
    """
    L=[]
    j=0
    while euler(j) <= ...
        if euler(j) >= ...
            L.append(euler(j))
        j+=1
    return L
```

Dans un fichier, que l'on appelera `prime.py` on met les fonctions : `euler` et `liste_prime_euler`

On souhaite verifier si la fonction `liste_prime_euler` renvoie bien des nombres premiers. On compare cette liste avec celle appelée `wiki`. Dans cette fontion, on teste si chaque élément de la liste L passée en argument est present dans la liste wiki des nombres premiers :

3.3 Compléter la fonction `verifie` :

```
def verifie(L):
    """retourne un booleen apres avoir testé tous les elements de L
    """
    wiki =[41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
    verif = True
    for n in L:
        if not n in wiki:
          ...
    return verif
```

3.4 Ecrire un test d'assertion, directement dans la fonction `liste_prime_euler`, pour vérifier si TOUS les éléments de la liste sont bien des nombres premiers, avant de retourner L. Le test utilisera le resultat de la fonction `verifie`.

```
def liste_prime_euler(a,b):
    L=[]
    j=0
    while euler(j) <= ...
        if euler(j) >= ...
            L.append(euler(j))
        j+=1

    assert ...
    return L
```

Une autre manière de vérifier si la liste L est correcte, c'est de programmer un test unitaire sur la fonction `liste_prime_euler` à l'aide du module `unittest`. On souhaite vérifier que chacune des valeurs calculée par la formule d'Euler est bien présente dans la liste des nombres premiers.

On met les instructions relatives à ce test dans un fichier appelé `main.py`. Le script de `main.py` est en partie reproduit ici :

```
import unittest
import prime


class PrimeTestCase(unittest.TestCase):
  """Test pour prime.py"""



  def test_euler(self):
    """verifie si les paramètres de liste_prime_euler
    sont dans la liste wiki
    """
    wiki =[41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
    premiers = prime.liste_prime_euler(40,100)
    for i in premiers:

        .....

if __name__=='__main__':
  unittest.main()
```

3.5 Compléter la fonction `test_euler` à l'aide de la fonction `assertIn(a, b)` du module unittest. Cette fonction renvoie True si a est dans la liste b. (voir les exemples du cours).