

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 :

```
1 def divide(a,b):  
2     return a/b
```

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

```
1 ZeroDivisionError      Traceback (most recent call last)  
2 <ipython-input-30-8044e08140eb> in <module>  
3 ----> 1 print(divide(10,0))  
4  
5 <ipython-input-29-8590a608aa42> in divide(a, b)  
6      1 def divide(a,b):  
7 ----> 2         return a/b  
8  
9 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 `divide`. En particulier, on ne sait pas dans quel ordre doivent être mis les arguments. On ne sait pas si on peut écrire `division(10,0)` ou bien `division(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 :

```
1 while True:  
2     try:  
3         x = int(input("Please enter a number: "))  
4         break  
5     except ValueError:  
6         print('erreur de type ValueError')  
7     except SyntaxError:  
8         print('erreur de type SyntaxError')  
9     else:  
10        print('erreur d'un autre type')
```

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

```
1 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 :

```

1 """retourne True si le nombre N passé en argument est premier False sinon
2 Param:
3 N : int
4         nombre à tester
5 Return:
6 True ou False
7 """

```

Et d'une fonction `liste_prime1` donnée ici :

```

1
2 def liste_prime1(a,b):
3     """retourne une liste de nombre premiers compris entre a et b
4     """
5
6     L=[]
7     for j in range(a,b+1):
8         if prem1(j):
9             L.append(j)
10
11     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 Ecrire le script de la fonction `liste_prime_euler` qui crée une liste de nombres premiers compris entre les bornes a et b à partir de cette formule.

Dans un même fichier, que l'on appellera `prime.py` on met les fonctions :

- `prem1` de l'exercice précédent
- `liste_prime1` de l'exercice précédent
- `liste_prime_euler`

On souhaite vérifier si la fonction `liste_prime_euler` renvoie bien tous les nombres premiers entre 40 et 100. On suppose que la fonction `prem1` donne les bons résultats.

3.2 Ecrire un test d'assertion, directement dans la fonction `liste_prime_euler`, pour vérifier si celle-ci calcule bien la bonne liste, avant de la renvoyer.

On souhaite programmer un test unitaire sur la fonction `liste_prime_euler` pour 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 :

```
1 import unittest
2 import prime
3
4 class PrimeTestCase(unittest.TestCase):
5     """Test pour prime.py"""
6
7     wiki =[41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
8
9     def test_euler(self):
10         """verifie si les paramètres de liste_prime_euler
11         sont entiers positifs
12         """
13         premiers = prime.liste_prime_euler(40,100)
14         for i in premiers:
15             # à compléter
16
17
18
19 if __name__ == '__main__':
20     unittest.main()
```

3.3 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`.