Rappel :

Lorsque vous créez une fonction, vous devez la documenter. C'est le rôle du docstring. Le docstring se place juste après la création de la fonction par def. Il commence et termine par **trois guillemets**.

Le docstring doit décrire le rôle de la fonction, puis les paramètres passés en arguments (type et rôle), ainsi que le type de ce qui est retourné.

Exercice 1

La mesure d'un angle peut être donnée en degrés ou en radian. L'unité de calcul naturelle pour les angles est le radian. La bibliothèque math ne sait donc calculer le cosinus que d'un angle donné en radian. ma\_fontion1, convertit des degrés en radians car pi radians correspondent à 180°. Modifier les noms des fonctions, variables et créer le docstring selon les normes énoncées plus haut.

import math

def ma\_fonction1(x):

return(x\*math.pi/180)

def ma\_fonction2(x):

return(math.cos(ma\_fonction1(x)))

Petite remarque, on importé le module math de la bibliothèque standard (la bibliothèque de Python). Ci-dessus, on l'a juste préparé et on appelle les fonctions qu'il contient au fur et à mesure des besoins (exemple math.cos). Ci-dessous, j'importe directement ce que je vais utiliser. Les fonctions sont chargées en mémoire et donc directement connues dans mon programme (je tape cos directement, plus math.cos). A noter si je remplace par from math import \*, alors je charge dans la mémoire de mon programme toutes les fonctions qui existent dans math...donc je surcharge inutilement.

from math import pi,cos

#from math import \*

def ma\_fonction1(x):

return(x\*pi/180)

def ma\_fonction2(x):

return(cos(ma\_fonction1(x)))

Exercice 2

Comprendre ce que fait le code suivant et corriger tous les problèmes de spécification.

def fonction(a,b):

return(a\*100/b)

Exercice 3

Comprendre ce que fait le code suivant et corriger tous les problèmes de spécification.

import math

def fonction1(a,b,c,d):

return(a+b+c+d)

def fonction2(a,b,c,d):

return(fonction1(a,b,c,d)/4)

Exercice 4

a) Comprendre ce que fait le code suivant et corriger tous les problèmes de spécification  
b) Que dois-je taper pour avoir l'aire d'un rectangle de côté 3 et 4, agrandit 10 fois?

c) Que dois-je taper pour que le programme affiche le volume d'un pavé droit de mesure 5;7;6 cm, agrandit 3 fois?

import math

def fonction1(a,b):

coefficient=10

c=coefficient\*coefficient\*a\*b

print("L'aire d'un rectangle de mesure",a,"et",b,"qui subit un agrandissement de coefficient",coefficient,"est",c)

#volume d'un parallélépipède de coté a,b,c qui subit un agrandissement de coefficient d

def fonction2(a,b,c,d):

return(a\*b\*c\*d\*d\*d)

Exercice 5

Comprendre ce que fait le code suivant et corriger tous les problèmes de spécification.

def fonction1(a):

print("1 - jouer à la bataille navale")

print("2 - jouer au puissance 4")

print("3 - quitter")

a=input("Taper votre choix 1 ou 2 ou 3:")

while a!="1" or a!="2" or a!="3":

a=input("Taper votre choix :")

return(a)

Exercice 6

Complétez-le en ajoutant le docstring et les commentaires.

def division\_euclidienne(a, b):

""" Ecrire la spécification de la fonction

Arguments

---------

Returns

-------

"""

# Ajouter des commentaires pour expliquer

if b <= 0 or a < 0 or type(a) != int or type(b) != int:

return -1

if a == 0:

return (0, 0)

# mes commentaires ici sur une ligne

r = a

q = 0

while (r >= 0):

r = r - b

q = q + 1

return (q-1, r+b)

# Tests

def test\_division\_euclidienne():

# Ajouter des commentaires pour expliquer

# quelles sont les vérifications effectuées

assert division\_euclidienne(10, 2) == (5, 0)

assert division\_euclidienne(2, 10) == (0, 2)

assert division\_euclidienne(37, 3) == (12, 1)

# Verifications d'erreurs

assert division\_euclidienne(-10, 7) == -1

assert division\_euclidienne(10, -7) == -1

assert division\_euclidienne(10.3, 4) == -1

assert division\_euclidienne(11, 3.5) == -1

assert division\_euclidienne(3, 0) == -1

# ...

assert division\_euclidienne(0, 3) == (0, 0)

assert division\_euclidienne(0, 0) == -1

# Lancement du test

test\_division\_euclidienne()

Exercice 7

Créer et donner le doctring d’une fonction appartient() qui indique si un élément est présent dans une liste, et qui passe le jeu de tests suivants:

def test\_appartient():

assert appartient(2, [5, 4, 1, 2, 3]) == True

assert appartient(6, [5, 4, 1, 2, 3]) == False

Compléter le jeu de tests avec des cas limites: liste vide, types de données différents…

Exercice 8

1. Rechercher dans la documentation du module random comment créer un dé numérique qui renvoie de façon équiprobable les valeurs entières de 1 à 6. Tester la fonction pour vérifier son fonctionnement.
2. Rechercher dans la documentation du module math comment calculer le logarithme en base 2 d'un nombre. Tester la fonction pour vérifier son fonctionnement: p.ex log2​(16)=4; log2​(256)=8. Expliquer ce que calcule cette fonction.
3. Rechercher dans la documentation de la librairie pandas plusieurs méthodes permettant d'itérer sur les valeurs d'une Dataframe.