TD 2 Prog.

5. Donner les préconditions des fonctions dont les spécifications sont les suivantes :

a. renvoie le périmètre du quadrilatère dont les longueurs des côtés (des entiers), sont passées en paramètre
b. renvoie la moyenne des 2 notes passées en paramètre

a. def perimetre_quad(c1, c2, c3, c4) -> float :
"""
c1, c2, c3, c4 > 0, c1, c2, c3, c4 : float.

b. def moyenne(note1 : float, note2 : float) -> float :
"""
note1, note2 ⩾ 0
"""

6. Décrivez le résultat des expressions suivantes ainsi que l'état de la mémoire durant leur évaluation :

```
def f(x:int, y:int)->int:
    return x + y + 1
```

```
def g(x:int, y:int)->int:
    res = x
    res = res + y
    return res + 1
```

```
DELTA = 1
def h(x:int, y:int)->int:
    res = x + y
    return res + DELTA
```

a. f(2, 3+4)
b. g(2, 3+4)
c. h(DELTA+1. 3+4)

a.

	x	y
11	2	7

f(2, 3+4) renvoie 10.

b.

	x	y	res
10	2	7	∅
11	2	7	2
12	2	7	9

g(2, 3+4) renvoie 10

c.

	DELTA	x	y	res
11	1	∅	∅	∅
12	1	2	7	9

h(DELTA+1, 3+4) renvoie 10.

7. Que pensez-vous du code suivant ?

```
x = 3
def f(x : int, y : int) -> int:
    return x + y + 1
def g(z : int) -> int:
    x = z**2
    return x + 12
```

Il y a un problème concernant l'appellation des variables. Dans les deux fonctions f et g, il y a une variable locale x, alors qu'il y a également une variable globale x. Ainsi si le code fonctionne, il est mauvais à cause de cette ambiguité.

8. Donner le verdict associé aux tests suivants :

```
def f(x : int , y : int) -> int:
    """Renvoie le reste de la division entière de x^2 par y

    Précondition : y non nul
    Exemple(s) :
    $$$ f(6, 5)
    1
    $$$ f(5, 0)
    1
    $$$ f(7, 5)
    1
    """
    return (x**2) % y
```

Test 1 OK

Test 2 non, on lève une exception.

Test 3 non, le résultat n'est pas 1.

9. Quels tests vous semblent pertinents pour les fonctions et opérateurs Python suivants :

a. `abs` qui renvoie la valeur absolue de son paramètre
b. opérateur `%`
c. `len` qui renvoie la longueur de la chaîne passée en paramètre (on considérera uniquement le cas où le paramètre est de type `str`)

a. $$$ abs(0) $$$ abs(-1) $$$ abs(3)
0 1 3

b. $$$ 5%5 $$$ -31%2 $$$ 0%2
0 1 0

c. $$$ len("") $$$ len("Hello, World!") $$$ len("2")
0 13. 1

10. Écrire la documentation complète de la fonction `double_abs` qui prend en paramètre un entier `x` et qui renvoie le double de la valeur absolue de `x`.

La fonction double_abs prend un paramètre entier x et elle renvoie le double de la valeur absolue de x

Préconditions : /

$$$ double_abs(0)
0
$$$ double_abs(-2)
4
$$$ double_abs(-7)
14.

5 Écriture de fonctions

Le but de l'exercice est de traiter un problème complexe en le décomposant et en le codant avec des fonctions. Le problème est : on dispose d'un entier positif à 4 chiffres (compris entre 1000 et 9999 inclus), par ex `1234`. On veut calculer une chaîne de caractères de taille 4 qui masque les chiffres des centaines et des dizaines de cet entier. Par exemple le masquage de `1234` produira la chaîne de caractères `'1**4'`[2].

11. Questions à se poser avant d'attaquer le problème, en s'appuyant sur l'exemple de l'énoncé :
    - quelle est la donnée principale ? son type ? quels opérateurs connus s'appliquent aux données de ce type ?
    - quel est le type du résultat à calculer pour résoudre le problème ?
    - quelles opérations a-t-on besoin d'appliquer à la donnée principale pour résoudre le problème ? Dispose-t-on d'opérateurs prédéfinis qui réalisent ces opérations de manière directe ?
    - de même : quelles opérations sont nécessaires pour construire le résultat ? Dispose-t-on d'opérateurs prédéfinis qui réalisent ces opérations de manière directe ?
12. Suggérer une décomposition de ce problème en sous-problème pouvant être codés par des fonctions.
13. Écrire une fonction `masque` qui prend en paramètre un entier comme décrit dans l'énoncé et renvoie une chaîne de caractères correspondant à son masquage, ainsi que les fonctions intermédiaires prévues précédemment. Par exemple `masque(1234)` vaut `'1**4'`.

11. La donnée est un nombre entre 1000 et 9999 et les opérateurs usuels sur les entiers s'appliquent à cette donnée.
Le type est une chaîne de caractère (= du retour de la fonct°).

12. Il faut : transformer le int en str
récupérer le i-ème élément d'un str.
concaténer des chaines de caractères

13.
```
def int_to_str(number:int)->str:
    '''
    Renvoie la conversion du type int vers le type str
    Précond: /.
    '''
    return str(number).

def recuperer_caractere(chaine:str, i:int)->str:
    '''
    Renvoie chaine[i].
    Précond: 0<i<len(chaine).
    '''
    return chaine[i].
def concat(chaine1:str, chaine2:str)->str:
    '''
    Renvoie chaine1 suivi de chaine2 dans un seul string.
    Précond. -
    '''
    Return chaine1+chaine2.
```

```
def masque(number: int) -> str:
    """
    Renvoie la chaine de caractères str(number) en
    cachant les chiffres du milieu.

    Précond: 1000 ≤ number ≤ 9999.
    """
    nombre_transforme = int_to_str(number)
    premier_chiffre = recuperer_caractere(nombre_transforme, 0)
    dernier_chiffre = recuperer_caractere(nombre_transforme, 3)
    return concat(premier_chiffre, concat("**", dernier_chiffre))
```