
DEFINITIONS, APPELS ET DOCUMENTATION

Exercice n°1 : Fonctions natives et documentation

1) Faites apparaître la documentation de la fonction `pow()`

a- A quoi sert-elle ? Testez-la avec une exemple de votre choix pour le vérifier.

b- Combien de paramètres au minimum doivent être utilisés ?

2) Faites apparaître la documentation de la fonction `abs()`

a- A quoi sert-elle ? Testez-la avec une exemple de votre choix pour le vérifier.

b- Combien de paramètre(s) doivent être utilisés ?

Exercice n°2 : Quelques fonctions personnalisées


Pour cet exercice, créez préalablement un script nommé `fonctions_persos.py`. Pour chacun des cas suivants vous définirez correctement une fonction en n'oubliant pas de :

- choisir un nombre de paramètres adaptés
- nommer les fonctions et les paramètres de manière non ambiguë
- de créer une docstring associée et de tester.

1) Définir une fonction retournant le carré d'un nombre

2) Définir une fonction retournant le périmètre d'un rectangle

3) Définir une fonction retournant la moyenne de 3 notes

4)  Définir une fonction retournant la moyenne pondérée de 2 notes

Exercice n°3 : IMC

1) Créer un nouveau script nommé `imc.py`.

2) En reprenant la définition vue lors de la séance précédente indiquez combien de paramètres sont nécessaires à la définition d'une fonction permettant de calculer l'IMC.

3) Créez la fonction `calcul_imc` correspondante en choisissant des noms de paramètres explicites. Vous n'oublierez pas de renseigner la docstring

4) Après avoir sauvegardé votre script, testez-la fonction avec les exemples suivants :

```
>>> calcul_imc(95, 1.81)
28.997893837184456
>>> calcul_imc(140, 2.04)
33.64090734332949
```

5) On souhaite améliorer cette fonction en donnant une valeur arrondie au dixième du résultat.

Ecrivez une fonction `calcul_imc_bis` en utilisant la fonction `round` vue en cours (reprenez la documentation associée si nécessaire)

```
>>> calcul_imc_bis(95, 1.81)
29.0
>>> calcul_imc_bis(140, 2.04)
33.6
```

Exercice n°4 : année-lumière

(source de l'exercice : [cours info L1S1 FIL](#))

L'**année-lumière** (*al*) est définie comme la distance parcourue par un photon (ou plus simplement la lumière) dans le vide, en une année julienne (soit 365,25 jours, ou 31 557 600 secondes).

 Quelques distances en $al^{[1]}$

- 1) Sachant que la vitesse de la lumière dans le vide est une constante fixée à 299 792 458 m/s, calculez à combien de kilomètres correspondent une année-lumière à l'aide d'instructions Python.
- 2) Écrivez une fonction qui convertit une distance exprimée en années-lumière en une distance exprimée en kilomètres.
- 3) Sachant que Proxima Centauri, l'étoile la plus proche du Système solaire, se trouve à 4,22 années-lumière de la Terre, à combien de km de la Terre cette étoile se trouve-t-elle ?

UTILISATION DU MODULE MATH

Exercice n°5 : Théorème de Pythagore

1) Définissez la fonction `hypotenuse`. Elle acceptera deux paramètres notés `a` et `b` correspondant aux longueurs des 2 plus petits côtés d'un triangle rectangle (exprimées avec la même unité). La valeur de retour de cette fonction sera la longueur de l'hypoténuse.

2) Enregistrez le script sous le nom `hypotenuse.py`

3) Vérifiez votre fonction à l'aide des exemples suivants:

```
>>> hypotenuse(2, 3)
3.605551275463989
>>> hypotenuse(5, 2)
5.385164807134504
```

- 4) Le module `math` possède une fonction nommée `hypot`.
Importez-la, puis recherchez dans sa documentation son mode d'utilisation.
Que constatez-vous ?

Exercice n°6 : Autour du cercle

- 1) Créer un nouveau script nommé `cercle.py`.
Importez-y `pi` provenant de la bibliothèque `math`.

Vous pouvez évaluer sa valeur :

```
>>> pi
3.141592653589793
```

- 2) Définissez la fonction `perimetre`. Elle acceptera un seul paramètre noté `rayon` correspondant au rayon d'un cercle.

La valeur de retour de cette fonction sera le périmètre de ce cercle.

Vérifiez votre fonction à l'aide des exemples suivants:

```
>>> perimetre(2)
12.566370614359172
>>> perimetre(100)
628.3185307179587
>>>
```

- 3) Toujours dans le même script, définissez la fonction `aire`. Elle acceptera un seul paramètre noté `rayon`.
La valeur de retour l'aire du disque correspondant.

Vérifiez votre fonction à l'aide des exemples suivants:

```
>>> aire(1)
3.141592653589793
>>> aire(5)
78.53981633974483
>>>
```

- 4) 🗑️ Après avoir sauvegardé votre script précédent, fermez le fichier puis ouvrez une nouvelle fenêtre et enregistrez un nouveau script pour l'instant vierge sous le nom `test_module.py` dans le même dossier que le script précédent

`cercle.py` peut être utilisé comme un module indépendant. Essayez d'importer le module `cercle` dans votre nouveau script et utilisez les fonctions précédemment définies.

Exercice n°7 : Des fonctions en série

Une fonction peut appeler d'autres fonctions :

1) Définir une fonction `cube` retournant le cube d'un nombre

2)


a- Définir une fonction `volume_sphere` retournant le volume d'une sphère

b- Modifiez cette fonction en utilisant la fonction `cube` précédemment définie

UTILISATION DU MODULE RANDOM

Exercice n°8 : Lancer de dés

1) Beaucoup de jeu de rôles (JDR) utilisent des dés spéciaux avec un nombre de faces adapté.

 Dé 20 faces[^2]

Créez une fonction `dice_20` sans paramètre simulant le lancer d'un dé à 20 faces (le nombre le plus petit étant 1 et les valeurs allant de un en un)

remarque : lors de la définition d'une fonction sans paramètre les parenthèses sont tout de même obligatoires:

```
def fonction_sans_parametre():  
    .....
```

2) Créez une fonction `dices_roll` (sans paramètre également) simulant le lancer simultané de 3 dés 20. Cette fonction retourne uniquement la somme du lancer : **vous utiliserez bien évidemment la fonction précédente**.

Exercice n° 9 : QCM

Indiquez à chaque fois **la bonne réponse** (l'utilisation de l'ordinateur est interdit):

Question 1

On considère la fonction suivante :

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

Quelle est la valeur de retour pour `fonction_mystere(4, 2)`?

- ☐ 0

- ☐ 2.0
- ☐ 4
- ☐ 2

Question 2

On considère la fonction suivante :

```
def fonction_mystere_bis(a):  
    b = 3  
    return b + a / 3
```

Quelle est la valeur de retour pour `fonction_mystere_bis(9)`?

- ☐ 4
- ☐ 4.0
- ☐ 6.0
- ☐ 6

Question 3

On considère le code suivant :

```
from math import sqrt  
  
def pythagore(a, b):  
    """ Renvoie la longueur de l'hypoténuse """  
    return nom_fonction(a * a + b * b)
```

Par quoi remplacer `nom_fonction` ?

- ☐ `math.sqrt`
- ☐ `sqrt`
- ☐ `sqrt.math`
- ☐ `math`


Question 4

On considère la fonction `f()` suivante:

```
def f(a, b, c, d):  
    a = b  
    c = d  
    return b ** b + c * d
```

Quelle est la valeur de retour pour `f(3, 2, 1, 4)`?

- ☐ 20
- ☐ 28

-  8
 -  9
-

Sources

[^1]: [Light year](#) by Bob King

[^2]: [20-sided dice](#), Photo by Fantasy on the English Wikipedia project