TP 4 : Fonction

Description

Objectifs

✔ Définition et utilisation de fonctions

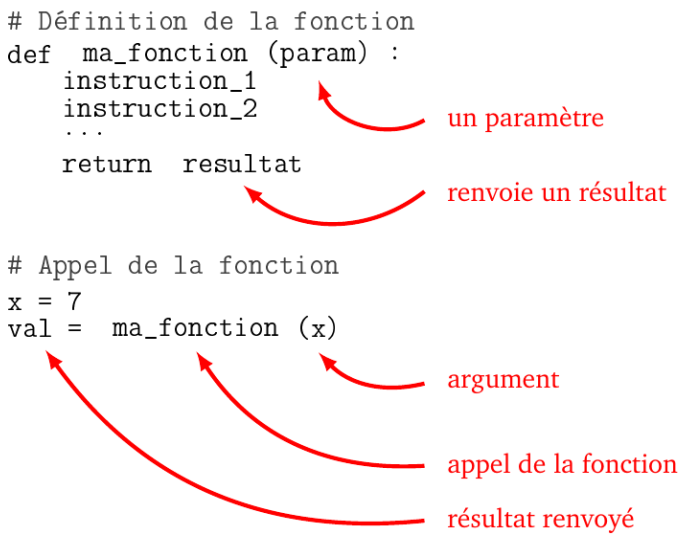
Prérequis

- Variables
- Boucles itératives

Memo de cours

i Il y a deux processus distincts dans l'utilisation des fonctions :

- La définition : étape pendant laquelle on définit les instructions à éxécuter par la fonction.
- Les appels : l'utilisation de la fonction avec des arguments particuliers.



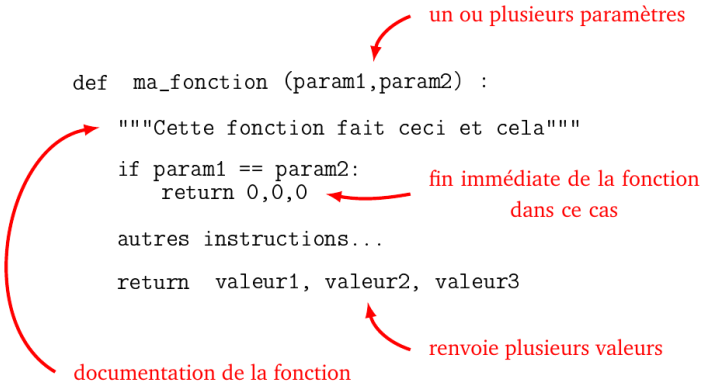

```
# Définition de la fonction
def  ma_fonction (param) :
    instruction_1
    instruction_2
    ...
    return  resultat

# Appel de la fonction
x = 7
val =  ma_fonction (x)
```

Source : Apprendre Python au lycée

Définition d'une fonction

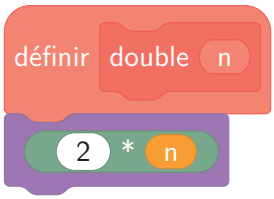
Pour **déf**inir une fonction on utilise le mot clé `def`.
Les parenthèses qui suivent le nom de la fonction accueillent les paramètres si nécessaire.
Les instructions de la fonction sont regroupées dans un bloc indenté.
Le mot `return` (optionnel) indique la fin de la fonction et précède les valeurs renvoyées par la fonction.

def nom_fonction **(** parametre_1**,** parametre_2 **):**
    bloc instructions
    return resultat



Source : Apprendre Python au lycée

•• De Scratch à Python

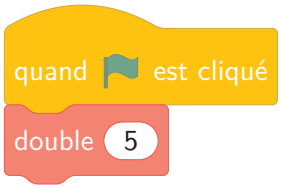



```
1  def double(n):
2      print(2*n)
```

•Appel d'une fonction

Pour utiliser une fonction il suffit de l'appeler par le nom qu'on lui a donné au moment de sa déclaration. Lors de cet appel, il faut lui fournir les arguments nécessaires.

•• De Scratch à Python




```
1  double(5)
```

•Paramètres obligatoires et paramètres optionnels

Il est possible de mettre des paramètres optionnels, pour cela il suffit de leur affecter une valeur lors de la déclaration de la fonction.

! Dans la délaration de la fonction, les paramètres optionnels doivent être en dernier

Exemple

```
1  def modulo(a,  b = 2):
2      mod = a % b
3      return mod
4
5  modulo(15)
6  modulo(15, 3)
```

— Le premier appel à la fonction `modulo`, c'est à dire `modulo(15)` renverra 1, car le second paramètre n'étant pas fourni, l'interpréteur pyhton prendra la valeur par défaut donc 2.

— Le second appel à la fonction `modulo`, c'est à dire `modulo(15, 3)` renverra 0, car le second paramètre est fourni, l'interpréteur pyhton renverra le reste de la division euclidienne de 15 par 3.

Exercices

Exercice 13 Discriminant

Définir une fonction `discriminant` qui prend en paramètre `a,b` et `c`, des nombres flottants représentant les coefficients d'un polynôme de second degré (ax^2+bx+c) et qui renvoie le discriminant Δ.

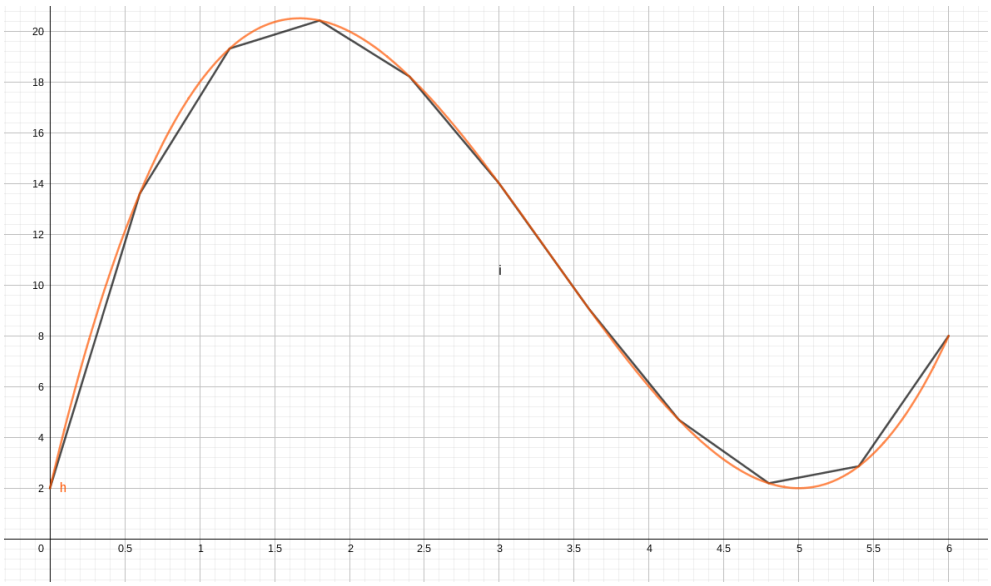
nom	`discriminant`
paramètre.s	`a, b, c` nombres flottants
retour.s	Nombre flottant
description	fonction qui renvoie le discriminant du polynôme de second degré $ax^2 + bx + c$.

Correction

```
def discriminant(a, b,c):
    delta = b**2 - 4*a*c
    return delta
```

Exercice 14 Longueur

✔ Objectif : Implémenter une méthode d'approximation de la longueur de la courbe représentative d'une fonction sur un intervalle borné en l'approchant par la somme des longueurs des segments ayant pour extrémités des points de la courbe.



1. Définir une fonction `f`

   La fonction `f` prend en paramètre x un nombre flottant et renvoie l'image de x par la fonction f définit par :

$$f(x) = x^3 - 10x^2 + 25x + 2$$

nom	`f`
paramètre.s	`x` un nombre flottant
retour.s	nombre flottant
description	fonction qui renvoie l'image de x par la fonction f.

Correction

```
def f(x):
    return x**3 - 10*x**2 + 25*x +2
```

2. Définir une fonction `distance(xA, yA, xB, yB)`.

   La fonction `distance` qui prend en argument les 4 nombres xA, yA, xB et yB, représentant les coordonnées des points $A(xA,\ yA)$ et $B(xB,\ yB)$ dans un repère othonormal, est renvoie la longueur AB.

nom	`distance`
paramètre.s	`xA, yA,xB,yB` nombres flottants
retour.s	Nombre flottant
description	fonction qui renvoie la distance entre les points $A(xA,\ yA)$ et $B(xB,\ yB)$.

Correction

```
from math import sqrt

def distance(xA,yA,xB,yB):
    return sqrt( (xB - xA)**2 + (yB - yA)**2)
```

3. Définir une fonction `longueur_courbe(f, a, b, n)` prend en paramètre `f` une fonction, `a` un nombre flottant, `b` un nombre flottant et `n` le nombre de segments utilisés pour l'approximation et renvoie la longueur.

nom	`longueur_courbe`
paramètre.s	`f` une fonction `a` la borne inférieure de l'intervalle `b` la borne supérieure de l'intervalle `n` le nombre de segments utilisés
retour.s	nombre flottant
description	fonction qui renvoie `True` si `n` admet une décomposition telle que le présente la conjecture de Goldbach et `False` sinon.

Correction

```
def longueur_courbe(f,a,b,n):
    pas = (b-a) / n
    longueur = 0
    for i in range(n):
        xA = a + i *pas
        xB = a + (i+1)*pas
        yA = f(xA)
        yB = f(xB)
        longueur += distance(xA,yA,xB,yB)
    return longueur
```

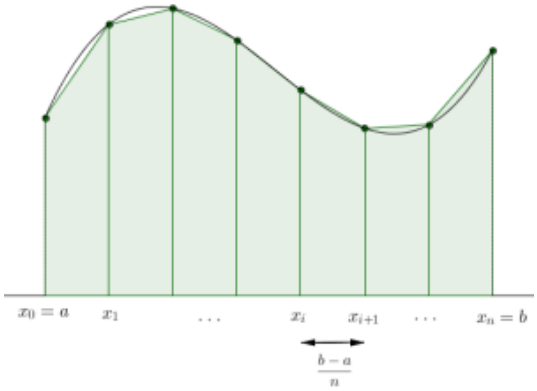
4. A l'aide de la fonction `longueur_courbe`, estimer la longueur de la courbe représentative de la fonction f sur l'intervalle $[\,0\,;\,6\,]$ avec 100 segments ?

Correction

```
print(longueur_courbe(f, 0, 6, 100))
```

Exercice 15 Aire sous la courbe

✔ Objectif : Implémenter la méthode des trapèzes pour calculer l'aire sous la courbe d'une fontion f continue positive (c'està dire l'aire de la surface comprise entre l'axe des abcsisses, les droites d'équation $x = a$ et $x = b$ et la courbe représentative de la fonction f).



1. Définir une fonction `aire_trapeze(xA, yA, xB, yB)`.
   La fonction `aire_trapeze` qui prend en argument les 4 nombres xA, yA, xB et yB (), renvoie l'aire d'un trapèze dont les coordonnées des sommets seraient $(xA\,;\,0), (xA\,;\,yA), (xB\,;\,0)$ et $(xB\,;\,yB)$.

nom	`aire_trapeze`
paramètre.s	`xA, yA, xB, yB` nombres flottants
retour.s	Nombre flottant
description	l'aire d'un trapèze dont les coordonnées des sommets seraient $(xA\,;\,0), (xA\,;\,yA), (xB\,;\,0)$ et $(xB\,;\,yB)$.

2. Définir une fonction `f`
   La fonction `f` prend en paramètre x un nombre flottant et renvoie l'image de x par la fonction f définit par :

$$f(x) = \frac{10}{5 + \mathrm{e}^{-2x+9}}$$

nom	`f`
paramètre.s	`x` un nombre flottant
retour.s	nombre flottant
description	fonction qui renvoie l'image de x par la fonction f.

3. Définir une fonction `aire_sous_courbe(f, a, b, n)` prend en paramètre `f` une fonction, `a` un nombre flottant, `b` un nombre flottant et `n` le nombre de trapèzes utilisés pour l'approximation et qui renvoie l'approximation de l'aire sous la courbe de f.

nom	`aire_sous_courbe`
paramètre.s	`f` une fonction `a` la borne inférieure de l'intervalle `b` la borne supérieure de l'intervalle `n` le nombre de trapèzes utilisés
retour.s	nombre flottant
description	fonction qui renvoie .

4. A l'aide de la fonction `aire_sous_courbe`, estimer la longueur de la courbe représentative de la fonction f sur l'intervalle $[\,0\,;\,6\,]$ avec 50 segments ?

Correction

```
import math

def aire_trapeze(xA, yA, xB, yB):
    return abs(0.5*(xB-xA)*(yA+yB))

def f(x):
    return 10/(5 + math.exp(-2*x+9))

def aire_sous_courbe(f,a,b,n):
    pas = (b-a)/n
    aire = 0
    for i in range(n):
        xA = a + i*pas
        yA = f(xA)
        xB = a + (i+1)*pas
        yB = f(xB)
        aire += aire_trapeze(xA, yA, xB, yB)
    return aire

if __name__ == "__main__":
    print(aire_sous_courbe(f,2,6,50))
```