Chapitre 5: Fonctions et portée d'une variable

Première NSI

Année scolaire 2020/2021



***** Introduction

Pour factoriser ou clarifier du code, on peut étendre le langage avec une fonction qui englobe un bloc

Lorsqu'on utilise une variable, on a besoin de contrôler les valeurs qu'elle peut prendre et des effets de bord qu'une modification de sa valeur peut engendrer dans le reste du programme. Cela va dépendre de la portée de la variable.

L'objectif de ce chapitre est d'apprendre à utiliser de manière pertinente des fonctions et de maîtriser les règles de portée d'une variable.

Fonctions

I.1 Définir une fonction



🔁 Définition 1

Lorsqu'on a besoin de réutiliser tout un bloc d'instructions, on peut l'encapsuler dans une **fonction**. On étend ainsi le langage avec une nouvelle instruction. Une fonction sert à factoriser et clarifier le code, elle facilite la maintenance et le partage. C'est un outil de **modularité**.

Pour déclarer une fonction, on définit son en-tête avec son nom et des paramètres formels d'entrée. Vient ensuite le bloc d'instructions, décalé d'une indentation et qui constitue le le corps de la fonction.

Fonction avec return

```
def mafonction(parametre1, parametre2):
   bloc d'instructions (optionnel)
   return valeur
```

Fonction sans return

```
def mafonction(parametre1, parametre2):
   bloc d'instructions (non vide)
```

Si le corps de la fonction contient au moins une instruction préfixée par le mot clef return alors l'exécution d'un return termine l'exécution du corps de la fonction et renvoie une valeur au programme principal. Si le return est dans une structure de contrôle (boucle, test), il est possible que le corps de la fonction ne soit pas entièrement exécuté, on parle de sortie prématurée.



Une fonction sans return s'appelle une **procédure**, elle modifie l'état du programme principal par **effet** de bord. En Python, une procédure renvoie quand même la valeur spéciale None au programme principal.

On exécute une fonction en substituant aux paramètres formels des valeurs particulières appelées paramètres effectifs. On parle d'appel de fonction, on peut l'utiliser comme une expression si une valeur est renvoyée ou comme une **instruction** s'il s'agit d'une procédure.

Par exemple une fonction carre qui prend en paramètre un nombre x et qui renvoie son carré, s'écrira :



🦰 Programme 1

```
def carre(x):
   return x ** 2
```

Une fonction peut prendre plusieurs paramètres. Par exemple une fonction carre distance origine (x,y) qui prend en paramètres deux nombres x et y et qui renvoie le carré de la distance d'un point de coordonnées (x, y) à l'origine d'un repère orthonormal, s'écrira :

🛂 Programme 2

```
def carre_distance_origine(x, y):
   return x ** 2 + y ** 2
```

Une fonction peut retourner un tuple de valeurs. Par exemple une fonction coord vecteur qui prend en paramètres quatre nombres xA, yA, xB, yB et qui retourne les coordonnées du vecteur lié dont les extrémités ont pour coordonnées (xA, yA) et (xB, yB), s'écrira:

🦰 Programme 3

```
def coord_vecteur(xA, yA, xB, yB):
   return (xB - xA, yB - yA)
```

Voici un exemple de fonction sans paramètres d'entrée, ni valeur de retour (il s'agit donc d'une procédure).

🦰 Programme 4

```
def nsi():
   print("Numérique et Sciences Informatiques")
```



Attention, return valeur retourne valeur qu'on peut capturer dans une variable alors que print (valeur) affiche valeur sur la sortie standard (l'écran par défaut) mais valeur ne peut alors être capturée dans une variable. On donne ci-dessous un extrait de console Python, où on a défini maladroitement une fonction cube avec un print à la place d'un return. On ne récupère pas la valeur de retour souhaitée mais None lorsqu'on appelle la fonction.

```
In [10]: def cube(x):
           print(x ** 3)
    . . . :
In [11]: cube(4)
64
In [12]: b = cube(5)
125
In [13]: b
In [14]: print(type(b))
<class 'NoneType'>
In [15]: print(b + 1)
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

Exercice 1

1.	L'Indice de Masse Corporelle se calcule par la formule IMC = $\frac{\text{masse}}{\text{taille}^2}$ où la masse est en kilogrammes et la taille en mètres. Un IMC est considéré comme normal s'il est compris entre 18,5 et 25. En dessous de 18,5, la personne est en sous-poids et au-dessus de 25 elle est en sur-poids.
	Écrire une fonction d'en-tête $imc(m, t)$ qui retourne la classification de l'IMC correspondant à une masse de m kilogrammes et une taille de t mètres : classe 0 pour sous-poids, 1 pour normal et 2 pour surpoids.
2.	a. Écrire une fonction max2(a, b) qui retourne le maximum de deux nombre a et b.

LAC <u>é</u> G «n balc			Boucles	Première N
>				
}				
}				
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		<b>b.</b> Écrire une fonction	on max3(a, b, c) qui retourne le maximum de tro	ois nombres a, b et c.
}				
$\geq$				
$\xi$				
$\leq$				
}				
}				
		Prin	cipaux opérateurs de comparaison de variables	
		v ==	y x est égal à y	
			y x est egal a y y x est différent de y	
			y x est strictement supérieur à y	
		x <	y x est strictement inférieur à y	
			y x est supérieur ou égal à y	
			y x est inférieur ou égal à y	_
		Princ	cipaux opérateurs sur des expressions booleenes	
		E and	F Vraie si E est Vraie ET F est Vraie	
		E or F	Vraie si E est Vraie OU F est Vraie	
		not E	Vraie si E est Fausse	_
💰 E	xer	cice 2 Fonctions de	e tests	
₹			umoinsun(a,b,c) qui renvoie un booléen indiqua	ant si l'un au moins
$\geq$		nombresa, bouces	t positif.	
₹				
₹				
₹		•••••		
€				
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		Écrire une fonction to sont positifs.	ous(a,b,c) qui renvoie un booléen indiquant si to	us les nombres a, b
Ş				
}				
\geq				
\leq				
\}		Écrire une fonction cr l'ordre croissant.	roissant(a,b,c) qui renvoie un booléen indiqua	nt si a, b, c sont d

1.	Ecrire une fonction aumoinsun(a,b,c) qui renvoie un booléen indiquant si l'un au moins des nombres a, b ou c est positif.
2.	Écrire une fonction tous (a,b,c) qui renvoie un booléen indiquant si tous les nombres a, b, c sont positifs.

3. Écrire une fonction croissant(a,b,c) qui renvoie un booléen indiquant si a, b, c sont dans l'ordre croissant.

4.	Une année est bissextile si elle est divisible par 400 ou si elle n'est pas divisible par 100 et qu'elle est divisible par 4. Écrire une fonction <code>bissextile(a)</code> qui renvoie un booléen indiquant si l'année a est bissextile.

Entraînement 1

Écrire une fonction mention (note) qui prend en paramètre une note et renvoie la chaîne de caractères $\text{`R' si note < 10, `A' si 10} \leqslant note < 12, `AB' si 12 \leqslant note < 14, `B' si 14 \leqslant note < 16 \text{ et 'TB' sinon.}$

Utiliser des bibliothèques de fonctions

Méthode

On a parfois besoin d'utiliser des fonctions de Python qui ne sont pas chargées pas défaut. Ces fonctions sont stockées dans des programmes Python appelées modules ou bibliothèques. Par exemple le module math contient les fonctions mathématiques usuelles et le module random contient plusieurs types de générateurs de nombres pseudo-aléatoires.

Pour importer une fonction d'un module on peut procéder de deux façons :

```
#import du module de mathématique (création d'un point d'accès)
import math
#pour utiliser la fonction sqrt, on la préfixe du nom du module et d'un
    point
racine = math.sqrt(2)
```

Première façon

```
#import de la fonction sqrt du module math
from math import sqrt
racine = sqrt(2)
#Pour importer toutes les fonctions de math, ecrire
#from math import *
```

Deuxième façon



Pour obtenir de l'aide sur le module math dans la console Python, il faut d'abord l'importer avec import math puis taper help(math), mais le mieux est encore de consulter la documentation en ligne https://docs.python.org/3/. Sans connexion internet, on peut lancer en local le serveur web de documentation avec la commande python3 -m pydoc -b.

Principales fonctions du modules random:

Fonction	Effet
randrange(a,b)	retourne un entier aléatoire dans [a;b[
randint(a,b)	retourne un entier aléatoire dans [a;b]
random()	retourne un décimal aléatoire dans [0;1[
uniform(a,b)	retourne un décimal aléatoire dans [a;b]

3	Exei	rcice 3
***	1.	Écrire une fonction sommeDe(n) qui retourne la somme des résultats obtenus en lançant n dé à 6 faces.
\leq		
\lesssim		
§		
<u>}</u>		
	2.	Écrire une fonction urne () qui retourne le numéro de la boule tirée dans une urne qui contient cinq boules numérotées 1, trois boules numérotées 2 et deux boules numérotées 3.
§		
\ <u>\</u>		
⋛		
⋛		
₹		
ξ		
۶		

Entraînement 2

On lance un dé équilibré à six faces numérotées de 1 à 6.

Le code Python ci-dessous permet d'afficher la face supérieure du dé lors de dix lancers successifs d'un dé à 6 faces. On commence par importer la fonction randint du module random. Cette fonction prend deux paramètres : par exemple randint (1, 6) retourne un entier aléatoire compris entre 1 et 6, les bornes sont incluses.

```
from random import randint
for k in range(10):
```



```
print(randint(1, 6))
```

- 1. Écrire une fonction moyenneDe(n) qui retourne la valeur moyenne des faces obtenues sur un échantillon de n lancers.
- **2.** Écrire une fonction premier6() qui retourne le rang du premier 6 obtenu lorsqu'on lance successivement le dé.
- **3.** Écrire une fonction tempsAttente(n) qui retourne le temps d'attente moyen du premier 6 sur un échantillon de n lancers.

Méthode

Le module turtle est une implémentation en Python du langage Logo créé dans les années 1970 pour l'enseignement de l'informatique à l'école. Il est disponible dans la distribution standard de Python. En déplaçant une pointe de stylo qui peut être matérialisée par une tortue, on peut tracer des figures géométriques dans un repère cartésien dont l'origine est au centre de la fenêtre et dont l'unité par défaut est le pixel. Lorsqu'on déplace le crayon, il laisse une trace s'il est baissé ou pas de trace s'il est levé. Nous utiliserons les fonctions suivantes de turtle.

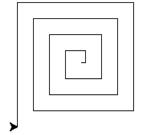
Syntaxe	Sémantique
goto(x,y)	déplace la tortue jusqu'au point de coordonnées (x, y)
penup()	lever le crayon
pendown()	baisser le crayon
setheading(angle)	choisir l'angle d'orientation de la tortue en degrés
forward(n)	avancer de n pixels selon l'orientation de la tortue
color("red")	choisir la couleur rouge (ou "black", "green", "blue")

On donne ci-dessous un programme permettant de tracer une spirale.

<mark>ệ</mark> Programme 5

```
from turtle import *

penup()
goto(0,0)
pendown()
c = 5
for i in range(4):
    for j in range(4):
        forward(c)
        c = 10 + c
        left(90)
exitonclick()
```



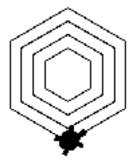


Exe	ercice 4
1.	. Écrire une fonction spirale1(n) qui permet de tracer une spirale constituée de n carrés déformés
2.	• Écrire une fonction $spirale2(n, m)$ qui permet de tracer une spirale constituée de n polygone déformés à m côtés.
2.	
	,

Entraînement 3

Écrire une fonction s

liers concentriques à Écrire une fonction spirale3(n, m) qui permet de tracer une spirale constituée de n polygones réguliers concentriques à *m* côtés.





Portée d'une variable



Définition 2

- Dans un code, la **portée d'une variable** définit les endroits du code où la variable est accessible.
- En Python, la portée d'une variable est lexicale c'est-à-dire qu'elle est définie par l'endroit où la variable est définie.
- En Python, une variable est **définie** dès qu'elle reçoit une valeur par une instruction d'**affectation**. C'est la dernière instruction d'affectation associée à un nom de variable qui détermine la portée de cette variable.
- Il existe deux grandes catégories de variables en Python :
 - Une variable définie dans une fonction est une variable locale à cette fonction, elle est accessible dans le bloc d'instruction de cette fonction et dans toutes les éventuelles fonctions qu'elle peut englober mais elle n'est pas visible dans tous les blocs qui englobent la fonction (programme principal ou fonctions englobantes).
 - Une variable définie dans le programme principal (pas dans une fonction) est une variable globale, elle est accessible dans l'ensemble du code.
- Un même nom de variable peut être utilisé dans une définition de variable par affectation à plusieurs niveaux d'imbrication de fonctions dans un même code.
 - Pour déterminer la portée d'une variable utilisée à un endroit fixé du code, on applique la règle **LEGB** pour Locale Englobante Globale Builtins. On recherche d'abord la variable dans la portée de la fonction locale, puis dans la portée d'une fonction englobante, puis dans le programme principal à l'extérieur de toute fonction et enfin dans le module builtins qui est importé par défaut.
- Il est possible d'enfreindre la règle **LEGB** en utilisant le mot clef **global** si on veut définir une variable dans une fonction avec une portée globale ou avec le mot clef nonlocal si on veut définir une variable dans une fonction englobée avec une portée dans la fonction englobante.
- La portée des variables en Python est très bien expliquée dans cette video :

https://d381hmu4snvm3e.cloudfront.net/videos/qYPPwG7tu2eU/SD.mp4



🦰 Programme 6

```
a = 1
def f():
    a = 734
    print(a)
f()
print(a)
```



🦰 Programme 7

```
a = 1
def f():
   a = a + 1
   print(a)
f()
print(a)
```



4	1
3	l
~	
5	
5	
<	
<	
<	
5	
<	
<	
5	
<	
<	
<	
<	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
5	
>	
<	

Exercice 5

1.	Que se passe-t-il lorsqu'on exécute les programmes 6 et 7 ci-dessus? Commenter.
2.	Que se passe-t-il lorsqu'on exécute le programme 8 ci-dessous? Commenter en précisant la portée des variables a, b et c lors de chaque exécution de l'instruction print (a, b, c).

<mark> Programme 8</mark>

```
a, b, c = 731, 734, 735
def f():
   b, c = 736, 737
   def g():
       c = 738
       print(a, b, c)
   g()
   print(a, b, c)
f()
print(a, b, c)
```



Exercice 6

Que se passe-t-il lorsqu'on exécute le programme 9 ci-dessous? Commenter en précisant la portée de la variable a lors des appels successifs des fonctions incremente 1(a), incremente 2() puis incremente 3().







ζ		
Ş		• • • • • • • • • • • • • • • • • • • •
ξ		
Ş		
Ş		
Ş		
Ş		
}	•	
ξ	· · · · · · · · · · · · · · · · · · ·	• • • • • • • • • • • • • • • • • • • •
>	•	

Programme 9

```
def incremente1(a):
   a = a + 1
def incremente2():
   global a
   a = a + 1
def incremente3():
   a = a + 1
a = 734
for k in range(10):
   incremente1(a)
print(a)
for k in range(10):
   incremente2()
print(a)
for k in range(10):
   incremente3()
print(a)
```