CHAPITRE 1

Calculs à la main

V. Le langage de programmation Python

a) Les différentes symboles opératoires

Opération	Écriture en Python	Résultat
Addition $7+3$	`7 + 3`	`10`
Soustraction $7-3$	`7 - 3`	`4`
Multiplication 7×3	`7 * 3`	`21`
Division $7\div 3$	`7 / 3`	`2.3333333333333335`
Division entière de 7 par 3	`7 // 3`	`2`
Puissance 7^3	`7 ** 3`	`343`
Valeur absolue $\lvert -7\rvert$	`abs(-7)`	`7`
Arrondir 2 chiffres après la virgule	`round(7/3, 2)`	`2.33`

Pour obtenir des fonctions ou des constantes particulières, on doit au préalable importer le module `math` une seule fois au tout début du programme :

Opération	Écriture en Python	Résultat
Le nombre π	`from math import pi` `2 * pi`	`6.283185307179586`
La fonction racine carrée $\sqrt{\ }$	`from math import sqrt` `sqrt(25)`	`5.0`

Remarque
Si on a besoin de plusieurs constantes ou fonctions spécifiques dans un programme on peut les importer sur une seule ligne en les séparant par une virgule :
`from math import pi, sqrt`

b) Affectation de variable

Effet	Écriture en Python	Résultat
Affecter à la variable `a` la valeur 6	`a = 6`	`a` contient 6
Affecter à la variable `b` la valeur contenu de `a` plus 3	`b = a + 3`	`a` contient 6, `b` contient 9
Augmenter la variable `a` de 1	`a = a + 1`	`a` contient 7, `b` contient 9

Remarque
En python, le symbole `=` n'a pas la même signification qu'en mathématiques, notamment, il n'est pas symétrique. Ainsi, si on a `a = 6`, on n'a pas `a = 6`.
Il faut donc traduire `=` par « prend pour valeur ».
On pourrait mentalement remplacer ce symbole par une flèche : `a` $\leftarrow$ `6`. Ainsi `6` $\leftarrow$ `a` n'a aucun sens.

c) Fonctions

Une fonction en Python prend un ou plusieurs arguments en entrée, exécute un travail sur ces arguments et renvoie un ou plusieurs résultats en sortie.

Syntaxe	Commentaires

`def nom_fonction(argument1, argument2, ...) :`
`  instructions éventuelles`
`  instructions éventuelles`
`  ...`
`  return résultat1, résultat2, ...`
	La première ligne se termine par :
	
	L'instruction `return` est nécessairement la dernière de la fonction.

Exemple (1)
La fonction suivante permet d'automatiser le calcul d'images de la fonction affine f d'expression $f(x) = 3x + 2$:

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

Ainsi `f(5)` renvoie 17, `f(10)` renvoie 32, ...

```
>>> f(10)
32
```

Exemple (2)
La fonction suivante permet d'automatiser le calcul de l'IMC (arrondi au centième) d'une personne connaissant sa taille en mètre et son poids en kg :

```
def imc(taille, poids) :
    calcul = poids / (taille**2)
    calcul = round(calcul, 2)
    return calcul
```

Ainsi pour une personne de 1,80 m pour 70 kg, son IMC est donné par `imc(1.8, 70)` :

```
>>> imc(1.8, 70)
21.6
```

d) Structure conditionnelle `Si ... Alors ... Sinon ...`

Les différentes structures conditionnelles :

Syntaxe	Commentaires
`if condition :` `  instruction`	Si la `condition` est vraie : alors on exécute l'`instruction`

`if condition :`
`  instruction1`
`else :`
`  instruction2`
	Si la `condition` est vraie :
	alors on exécute l'`instruction1`
	sinon
	on exécute l'`instruction2`

`if condition1 :`
`  instruction1`
`elif condition2 :`
`  instruction2`
`else :`
`  instruction3`
	Si la `condition1` est vraie :
	alors on exécute l'`instruction1`
	sinon si la `condition2` est vraie :
	alors on exécute l'`instruction2`
	sinon
	on exécute l'`instruction3`

Pour exprimer une condition on utilise les opérateurs suivants :

Syntaxe	Commentaires
`<`	est plus petit :
`<=`	est plus petit ou égal :
`>`	est plus grand :
`>=`	est plus grand ou égal :
`==`	est égal :
`!=`	n'est pas égal :

Exemple (1)
La fonction suivante permet de déterminer lequel des deux nombres donnés en argument est le plus grand :

```
def plusgrand(a, b) :
    if a > b :
        return a
    else :
        return b
```

Ainsi `plusgrand(5,4)` renvoie 5 :

```
>>> plusgrand(5,4)
5
```

et `plusgrand(-5,4)` renvoie 4 :

```
>>> plusgrand(-5,4)
4
```

Exemple (2)
Une attraction coûte un certain prix, mais les enfants de moins de 2 ans bénéficient d'une réduction de 90%, les enfants entre 2 et 12 d'une réduction de 60%, enfin les personnes de plus de 60 bénéficient eux d'une réduction de 20%. Les autres payent le plein tarif. La fonction suivante permet de déterminer le tarif en fonction du prix à payer de base et de l'âge du client :

```
def tarif(age, prix) :
    if age <= 2 :
        apayer = prix * 0.1
    elif age <= 12 :
        apayer = prix * 0.4
    elif age >= 60 :
        apayer = prix * 0.8
    else :
        apayer = prix
    return apayer
```

Ainsi `tarif(1,32)` renvoie 3.2 :

```
>>> tarif(1,32)
3.2
```

et `tarif(62,45)` renvoie 36.0 :

```
>>> tarif(62,45)
36.0
```

e) Boucle bornée : boucle `Pour`

Pour répéter des instructions un nombre connu de fois :

Syntaxe	Commentaires
`for i in range(n) :` `  instruction`	Pour `i` allant de 0 à `n-1` : on fait l'`instruction` (on l'a donc répétée `n` fois...)
`for i in range(n,m) :` `  instruction`	Pour `i` allant de `n` à `m-1` : on fait l'`instruction` (on l'a donc répétée `m-n` fois...)

Exemple
La fonction suivante permet de calculer la somme des `n` premiers entiers :

```
def somme(n) :
    S = 0
    for i in range(n+1):
        S = S + i
    return S
```

Ainsi `somme(5)` renvoie le résultat de $1+2+3+4+5$

```
>>> somme(5)
15
```

et `somme(100)` renvoie le résultat de $1+2+3+4+5+...+100$

```
>>> somme(100)
5050
```

f) Boucle non bornée : boucle `Tant que`

Pour répéter des instructions un nombre connu de fois :

Syntaxe	Commentaires
`while condition :` `  instruction`	Tant que la `condition` reste vraie : on fait l'`instruction` (on ne sait pas combien de fois on l'a répétée...)

Exemple
La fonction suivante permet de calculer la somme despremiers entiers jusqu'à ce qu'on dépasse un certain `seuil` :

```
def somme(seuil) :
    S = 0
    n = 1
    while S <= seuil:
        S = S + n
        n = n + 1
    return n, S
```

Ainsi `seuil(100)` renvoie le résultat de `(15, 105)`, autrement dit $1+2+3+4+5+...+15 \geq 100$
car $1+2+3+4+5+...+15 = 105$
(mais $1+2+3+4+5+...+14 < 100$).

```
>>> somme(100)
(15, 105)
```

g) Exercices sur le langage de programmation Python

Niveau 1 : Indiquer le résultat de chaque calcul exprimé en Python

```
3 + 2 * 5
```

```
5 ** 2
```

```
5 * 3 ** 2
```

```
5 / 2
```

```
5 / 2 * 3
```

```
3 + 4 * 7
```

```
2 ** 3
```

```
2 * 3 ** 2
```

Niveau 2 : Écrire l'expression algébrique correspondant à l'expression écrite en Python

```
2 + 3 * (x - 4)
```

```
5 * x ** 2 - 3 * x + 4
```

```
(3 * x - 1) * (2 * x + 1)
```

```
(2 * x + 1) / 3
```

```
1 / x + 2
```

```
x + 1 / (x + 1)
```

```
from math import pi
pi * r **  2 * h
```

```
from math import sqrt
sqrt(x**2 + 1)
```

```
from math import sqrt, pi
sqrt(V / (pi * h))
```

Niveau 3 : Donner la valeur demandée après exécution du programme

```
a = 1
b = 2*a + 3
c = b + 1
```

Valeur de `c`

```
a = 1
b = 2 + 3*a
c = b - a
```

Valeur de `c`

```
x = 4
x = x + 1
x = x * x
```

Valeur de `x`

```
cote = 3
P = 4 * cote
A = cote * cote
```

Valeurs de `A` et de `P`

```
a = 1
a = 2 + 3*a
a = a + 1
```

Valeur de `a`

```
a = 4
b = 5
a = a + b
b = a - b
```

Valeur de `b`

Niveau 4 : Donner la valeur demandée après exécution du programme

```
def f(x):
    return 3*x + 4
```

Que renvoie `f(-5)` ?

```
def g(t):
    return t**2 + 3*t - 4
```

Que renvoie `g(4)` ?

```
def aire(larg, longu):
    return larg * longu
```

Que renvoie `aire(3,8)` ?

```
def f(a, b):
    return a*b - a - b
```

Que renvoie `f(4, 3)` ?

```
def peri(l, L):
    return 2 * (l + L)
```

Que renvoie `peri(12, 3)` ?

```
def aire_tri(b, h):
    return b * h / 2
```

Que renvoie `aire_tri(10,8)` ?

Niveau 5 : Donner la valeur demandée après exécution du programme

```
def f(a):
    if a < 0:
        b = 2*a
    else :
        b = 3*a
    return b
```

Que renvoie `f(1)` ?
Que renvoie `f(-1)` ?

```
def f(a):
    if a >= 1:
        b = a - 1
    else :
        b = 2*a
    return b
```

Que renvoie `f(1)` ?
Que renvoie `f(-1)` ?

```
def f(a, b):
    if a > b:
        y = a - b
    else :
        y = b - a
    return y
```

Que renvoie `f(4, 7)` ?
Que renvoie `f(7, 4)` ?

```
def f(t):
    if t < 2:
        calcul = 2*t
    else :
        calcul = 2*(t - 2) + 1
    return calcul
```

Que renvoie `f(2)` ?
Que renvoie `f(-2)` ?

```
def f(x):
    if x != 0:
        return 1 / x
```

Que renvoie `f(0)` ?
Que renvoie `f(5)` ?

Niveau 6 : Boucle `Pour`

```
for i in range(4):
    calcul = 2*i
```

Donner toutes les valeurs prises par la variable `calcul`

```
for i in range(6):
    calcul = i**2
```

Donner toutes les valeurs prises par la variable `calcul`

```
calcul = 13
for i in range(4, 8):
    calcul = calcul + i
```

Donner toutes les valeurs prises par la variable `calcul`

```
def f(n):
    S = 0
    for i in range(n):
        S = S + 2*i + 1
    return S
```

Que renvoie `f(4)` ?

```
def f(n):
    S = 1
    for i in range(1, n):
        S = S * i
    return S
```

Que renvoie `f(5)` ?

```
def  f(a, n) :
    for i in range(n):
        a = a**2
    return a
```

Que renvoie `f(2,3)` ?

Niveau 7 : Donner la valeur demandée après exécution du programme

```
def mystere(n):
    s = 0
    for i in range(n):
        s = s + i
    return s
```

Que renvoie `mystere(5)` ?

```
def mystere(n):
    s = 0
    for i in range(n):
        s = s + i**2
    return s
```

Que renvoie `mystere(4)` ?

```
def mystere(n):
    s = 0
    for i in range(n):
        s = s + (-i)**i
    return s
```

Que renvoie `mystere(4)` ?

```
def mystere(n):
    s = 0
    for i in range(n):
        s = s + i
        if s == 6:
            s = s/2
    return s
```

Que renvoie `mystere(5)` ?

```
def mystere(n):
    s = 0
    for i in range(n):
        s = s + i**2
        if s == 3 :
            s = s + 1
        else :
            s = s - 1
    return s
```

Que renvoie `mystere(4)` ?

```
def mystere(n):
    s = 0
    for i in range(n):
        s = s + (-i)**i
        if s < 0:
            s = -s
        else :
            s = s + 2
    return s
```

Que renvoie `mystere(4)` ?

Niveau 8 : Boucle `Tant Que`

```
S = 1
while S < 10:
    S = S * 2
print(S)
```

Qu'affiche ce programme dans la console ?

```
S = 1
while S < 10:
    S = S * 2
print(S)
```

Qu'affiche ce programme dans la console ?

```
n = 0
S = 1
while S < 100:
    n = n + 1
    S = S * 5
print(n)
```

Qu'affiche ce programme dans la console ?

```
def mystere(a):
    while a < 13 :
        a = a + 3
    return a
```

Que renvoie `mystere(5)` et `mystere(13)` ?

```
def mystere(n):
    while n >= 5 :
        n = n - 5
    return n
```

Que renvoie `mystere(20)` et `mystere(7)` ?

```
def mystere(n):
    a = 1
    while a <= n :
        a = a * 2
    return a
```

Que renvoie `mystere(10)` et `mystere(25)` ?

VI. Correction des exercices

Niveau 1 : Indiquer le résultat de chaque calcul en Python

```
13
```

```
25
```

```
45
```

```
2.5
```

```
7.5
```

```
31
```

```
8
```

```
18
```

Niveau 2 : Écrire l'expression algébrique correspondant à l'expression écrite en Python

$2 + 3(x - 4)$

$5x^2 - 3x + 4$

$(3x - 1)(2x + 1)$

$\dfrac{2x + 1}{3}$

$\dfrac{1}{x} + 2$

$x + \dfrac{1}{x + 1}$

$\pi r^2 h$

$\sqrt{x^2 + 1}$

$\sqrt{\dfrac{V}{\pi h}}$

Niveau 3 : Donner la valeur demandée après exécution du programme

`a` vaut 1 et `b` vaut 5 donc `c` vaut 6

`a` vaut 1 et `b` vaut 5 donc `c` vaut 4

`x` vaut 25

`P` vaut 12 et `A` vaut 9

`a` vaut 6

`a` vaut 9 donc `b` vaut 4

Niveau 4 : Donner la valeur demandée après exécution du programme

`f(-5)` vaut -11

`g(4)` vaut 24

`aire(3,8)` vaut 24

`f(4, 3)` vaut 5

`peri(12, 3)` vaut 30

`aire_tri(10,8)` vaut 40

Niveau 5 : Donner la valeur demandée après exécution du programme

`f(1)` renvoie 3
`f(-1)` renvoie -2

`f(1)` renvoie 0
`f(-1)` renvoie -2

`f(4, 7)` renvoie 3
`f(7, 4)` renvoie également 3

`f(2)` renvoie 1
`f(-2)` ernvoie -4

`f(0)` ne renvoie rien
(heureusement car on ne peut diviser par 0)
`f(5)` renvoie 0.2

Niveau 6 : Boucle `Pour`

Les valeurs prises par la variable `calcul` sont successivement 0, 2, 4, 6.	Les valeurs prises par la variable `calcul` sont successivement 0, 1, 4, 9, 16, 25.	Les valeurs prises par la variable `calcul` sont successivement 13, 17, 22, 28, 35
`f(4)` renvoie 16	`f(5)` renvoie 24	`f(2,3)` renvoie 256

Niveau 7 : Donner la valeur demandée après exécution du programme

`mystere(5)` renvoie 10	`mystere(4)` renvoie 14	`mystere(4)` renvoie -23
`mystere(5)` renvoie 7	`mystere(4)` renvoie 12	`mystere(4)` renvoie 17

Niveau 8 : Boucle `Tant Que`

Ce programme dans la console 16.	Ce programme dans la console 2, 4, 8 puis 16.	Ce programme dans la console 3.
`mystere(5)` renvoie 14 et `mystere(13)` renvoie 13.	`mystere(20)` renvoie 0 et `mystere(7)` renvoie 2.	`mystere(10)` renvoie 16 et `mystere(25)` renvoie 32.