Premiers pas

Lance-toi dans la programmation ! Dans cette toute première activité, tu vas apprendre à manipuler des nombres, des variables et tu vas coder tes premières boucles avec `Python`.

Vidéo ■ Premiers pas - partie 1 - Lance-toi !
Vidéo ■ Premiers pas - partie 2 - Variable
Vidéo ■ Premiers pas - partie 3 - Fonctions
Vidéo ■ Premiers pas - partie 4 - Boucle pour
Vidéo ■ Installer Python
Vidéo ■ Démarrer Python et utiliser IDLE

Cours 1 (Nombres avec `Python`)**.**
Vérifie dans la console que `Python` fonctionne correctement, en tapant les commandes suivantes dans une console `Python` :

```
>>> 2+2
>>> "Bonjour le monde !"
```

Voici quelques instructions.

- **Addition.** `5+7`.
- **Multiplication.** `6*7` ; avec des parenthèses `3*(12+5)` ; avec des nombres à virgule `3*1.5`.
- **Puissance.** `3**2` pour $3^2 = 9$; puissance négative `10**-3` pour $10^{-3} = 0.001$.
- **Division réelle.** `14/4` vaut `3.5` ; `1/3` vaut `0.3333333333333333`.
- **Division entière et modulo.**
  — `14//4` vaut 3 : c'est le quotient de la division euclidienne de 14 par 4, note bien la double barre ;
  — `14%4` vaut 2 : c'est le reste de la division euclidienne de 14 par 4, on dit aussi « 14 modulo 4 ».

Remarque. Dans tout ce cours, on écrira les « nombres à virgule » sous la forme 3.5 (et pas 3,5). Le séparateur décimal est donc le point. En informatique les nombres à virgule sont appelés « nombres flottants ».

Activité 1 (Premiers pas)**.**

Objectifs : faire tes premiers calculs avec `Python`.

1. Combien y a-t-il de secondes en un siècle ? (Ne tiens pas compte des années bissextiles.)
2. Jusqu'où faut-il compléter les pointillés pour obtenir un nombre plus grand qu'un milliard ?
$$(1+2)\times(3+4)\times(5+6)\times(7+8)\times\cdots$$
3. Quels sont les trois derniers chiffres de
$$\underbrace{123456789\times 123456789\times\cdots}_{\text{7 occurrences de } 123456789} \ ?$$

4. 7 est le premier entier tel que son inverse a une écriture décimale périodique de longueur 6 :
$$\frac{1}{7} = 0.\underbrace{142857}\underbrace{142857}\underbrace{142857}\ldots$$
Trouve le premier entier dont l'inverse a une écriture décimale périodique de longueur 7 :
$$\frac{1}{???} = 0.00\underbrace{abcdefg}\underbrace{abcdefg}\ldots$$
Indication. L'entier est plus grand que 230 !
5. Trouve l'unique entier :
   - qui donne un quotient de 107 lorsque l'on effectue sa division (euclidienne) par 11,
   - et qui donne un quotient de 90 lorsque l'on effectue sa division (euclidienne) par 13,
   - et qui donne un reste égal à 6 modulo 7 !

Cours 2 (Travailler avec un éditeur)**.**
À partir de maintenant, il est préférable que tu travailles dans un éditeur de texte dédié à `Python`. Tu dois alors explicitement demander d'afficher le résultat :

```
print(2+2)
print("Bonjour le monde !")
```

Cours 3 (Variables)**.**
Variable. Une ***variable*** est un nom associé à un emplacement de la mémoire. C'est comme une boîte que l'on identifie par une étiquette. La commande « `a = 3` » signifie que j'ai une variable « `a` » associée à la valeur 3.
Voici un premier exemple :

```
a = 3  # Une variable
b = 5  # Une autre variable

print("La somme vaut",a+b)   # Affiche la somme
print("Le produit vaut",a*b) # Affiche le produit

c = b**a     # Nouvelle variable...
print(c)     # ... qui s'affiche
```

Commentaires. Tout texte qui suit le caractère dièse « # » n'est pas exécuté par `Python` mais sert à expliquer le programme. C'est une bonne habitude de commenter abondamment ton code.

Noms. Il est très important de donner un nom clair et précis aux variables. Par exemple, avec les noms bien choisis tu devrais savoir ce que calcule le code suivant :

```
base = 8
hauteur = 3
aire = base * hauteur / 2
print(aire)
print(Aire)  # !! Erreur !!
```

Attention ! `Python` distingue les majuscules des minuscules. Donc `mavariable`, `Mavariable` et `MAVARIABLE` sont des variables différentes.

Réaffectation. Imaginons que tu veuilles tenir tes comptes journaliers. Tu pars d'une somme $S_0 = 1000$, le lendemain tu gagnes 100, donc maintenant $S_1 = S_0 + 100$; le jour d'après tu rajoutes 200, donc $S_2 = S_1 + 200$; puis tu perds 50, donc au troisième jour $S_3 = S_2 - 50$. Avec `Python` tu peux n'utiliser qu'une seule variable `S` pour toutes ces opérations.

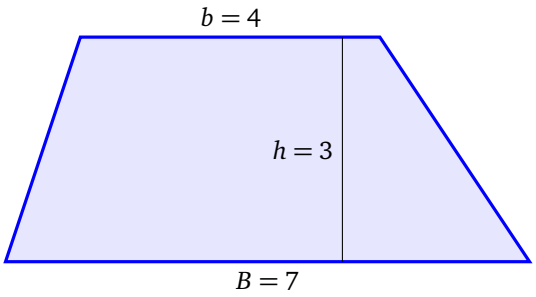
```
S = 1000
S = S + 100
S = S + 200
S = S - 50
print(S)
```

Il faut comprendre l'instruction « `S = S + 100` » comme ceci : « je prends le contenu de la boîte `S`, je rajoute 100, je remets tout dans la même boîte ».

Activité 2 (Variables)**.**

Objectifs : utiliser des variables !

1. (a) Définis des variables, puis calcule l'aire d'un trapèze. Ton programme doit afficher `"L'aire vaut ..."` en utilisant `print("L'aire vaut",aire)`.



   (b) Définis des variables pour calculer le volume d'une boîte (un parallélépipède rectangle) dont les dimensions sont 10, 8, 3.

   (c) Définis une variable `PI` qui vaut 3.14. Définis un rayon $R = 10$. Écris la formule de l'aire du disque de rayon R.

2. Remets les lignes dans l'ordre de sorte qu'à la fin x ait la valeur 46.

```
(1)  y = y - 1
(2)  y = 2*x
(3)  x = x + 3*y
(4)  x = 7
```

3. Tu places la somme de 1000 euros sur un compte d'épargne. Chaque année les intérêts sur l'argent placé rapportent 10% (le capital est donc multiplié par 1.10). Écris le code qui permet de calculer le capital pour les trois premières années.

4. Je définis deux variables par `a = 9` et `b = 11`. Je souhaite échanger le contenu de `a` et `b`. Quelles instructions conviennent de sorte qu'à la fin `a` vaut 11 et `b` vaut 9 ?

```
a = b
b = a
```

```
c = b
a = b
b = c
```

```
c = a
a = b
b = c
```

```
c = a
a = c
c = b
b = c
```

Cours 4 (Utiliser des fonctions).

- **Utiliser des fonctions de `Python`.**
  Tu connais déjà la fonction `print()` qui affiche une chaîne de caractères (ou des nombres). Elle s'utilise ainsi `print("Coucou")` ou bien à travers une valeur :

```
chaine = "Bonjour"
print(chaine)
```

  Il existe plein d'autres fonctions. Par exemple la fonction `abs()` calcule la valeur absolue : `abs(-3)` renvoie 3, `abs(5)` renvoie 5.
- **Le module `math`.**
  Toutes les fonctions ne sont pas directement accessibles. Elles sont souvent regroupées dans des *modules*. Par exemple le module `math` contient les fonctions mathématiques. Tu y trouves par exemple la fonction racine carrée `sqrt()` (*square root*). Voici comment l'utiliser :

```
from math import *

x = sqrt(2)
print(x)
print(x**2)
```

  La première ligne importe toutes les fonctions du module `math`, la seconde calcule $x = \sqrt{2}$ (en valeur approchée) et ensuite on affiche x et x^2.
- **Sinus et cosinus.**
  Le module `math` contient les fonctions trigonométriques sinus et cosinus et même la constante `pi` qui est une valeur approchée de π. Attention, les angles sont exprimés en radians.
  Voici le calcul de $\sin(\frac{\pi}{2})$.

```
angle = pi/2
print(angle)
print(sin(angle))
```

- **Décimal vers entier.**
  Dans le module `math` il y aussi des fonctions pour arrondir un nombre décimal :
  — `round()` arrondit à l'entier le plus proche : `round(5.6)` renvoie 6, `round(1.5)` renvoie 2.
  — `floor()` renvoie l'entier inférieur ou égal : `floor(5.6)` renvoie 5.
  — `ceil()` renvoie l'entier supérieur ou égal : `ceil(5.6)` renvoie 6.

Activité 3 (Utiliser des fonctions).

Objectifs : utiliser des fonctions de `Python` *et du module* `math`.

1. La fonction `Python` pour le pgcd est `gcd(a,b)` (sans le « p », pour *greatest common divisor*). Calcule le pgcd de $a = 10\,403$ et $b = 10\,506$. Déduis-en le ppcm de a et b. La fonction ppcm n'existe pas, tu dois utiliser la formule :
$$\text{ppcm}(a, b) = \frac{a \times b}{\text{pgcd}(a, b)}.$$
2. Trouve par tâtonnement un nombre réel x qui vérifie toutes les conditions suivantes (plusieurs solutions sont possibles) :
   - `abs(x**2 - 15)` est inférieur à `0.5`
   - `round(2*x)` renvoie 8
   - `floor(3*x)` renvoie 11

- `ceil(4*x)` renvoie 16

Indication. `abs()` désigne la fonction valeur absolue.

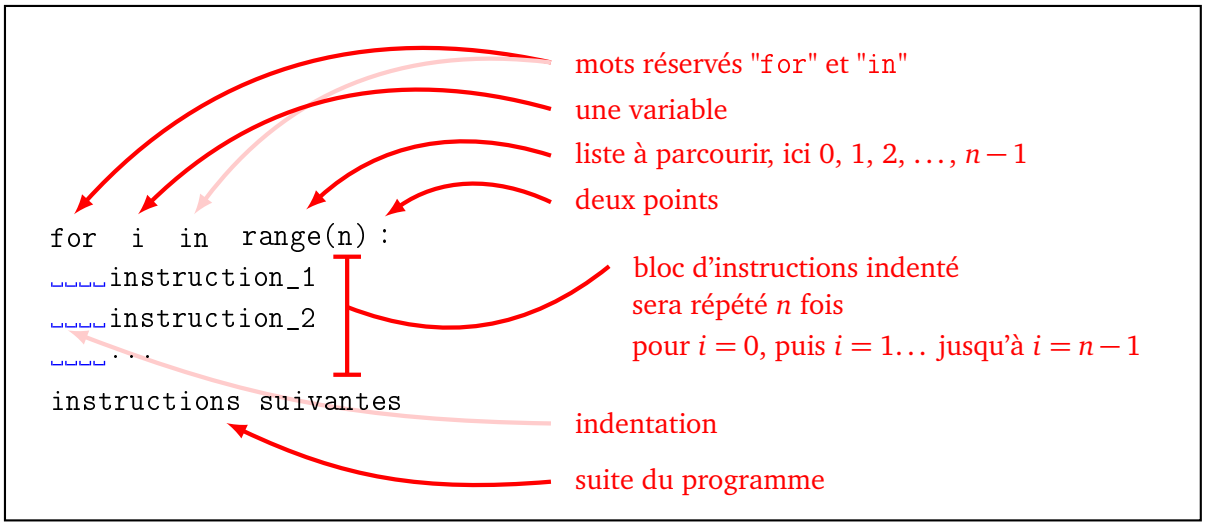
3. Tu connais la formule de trigonométrie

$$\cos^2\theta + \sin^2\theta = 1.$$

Vérifie que pour $\theta = \frac{\pi}{7}$ (ou d'autres valeurs) cette formule est numériquement vraie. (Ce n'est pas une preuve de la formule, car `Python` ne fait que des calculs approchés du sinus et du cosinus).

Cours 5 (Boucle « pour »)**.**

La boucle « pour » est la façon la plus simple de répéter des instructions.



Note bien que ce qui délimite le bloc d'instructions à répéter, c'est ***l'indentation***, c'est-à-dire les espaces placées en début de ligne qui décalent les lignes vers la droite. Toutes les lignes d'un bloc doivent avoir exactement la même indentation. Dans ce livre, nous choisissons une indentation de 4 espaces.
N'oublie pas les deux points en fin de la ligne de la déclaration du `for` !

- **Exemple de boucle « pour ».**
  Voici une boucle qui affiche les premiers carrés.

```
for i in range(10):
    print(i*i)
```

  La seconde ligne est décalée et constitue le bloc à répéter. La variable `i` prend la valeur 0 et l'instruction affiche 0^2 ; puis `i` prend la valeur 1, et l'instruction affiche 1^2 ; puis 2^2, 3^2...
  Au final ce programme affiche :

$$0, 1, 4, 9, 16, 25, 36, 49, 64, 81.$$

  Attention : la dernière valeur prise par `i` est bien 9 (et pas 10).
- **Parcourir une liste quelconque.**
  La boucle « pour » permet de parcourir n'importe quelle liste. Voici une boucle qui affiche le cube des premiers nombres premiers.

```
for p in [2,3,5,7,11,13]:
    print(p**3)
```

- **Sommes des entiers.**
  Voici un programme qui calcule

$$0+1+2+3+\cdots+18+19.$$

```
somme = 0
for i in range(20):
    somme = somme + i
print(somme)
```

Comprends bien ce code : une variable `somme` est initialisée à 0. On va tour à tour lui ajouter 0, puis 1, puis 2... On peut mieux comprendre cette boucle en complétant un tableau :

Initialisation : `somme`= 0

`i`	`somme`
0	0
1	1
2	3
3	6
4	10
...	...
18	171
19	190

Affichage : 190

- `range()`.
  - Avec `range(n)` on parcourt les entiers de 0 à $n-1$. Par exemple `range(10)` correspond à la liste `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`.
    Attention ! la liste s'arrête bien à $n-1$ et pas à n. Ce qu'il faut retenir c'est que la liste contient bien n éléments (car elle commence à 0).
  - Si tu veux afficher la liste des éléments parcourus, il faut utiliser la commande :
    `list(range(10))`
  - Avec `range(a,b)` on parcourt les éléments de a à $b-1$. Par exemple `range(10,20)` correspond à la liste `[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]`.
  - Avec `range(a,b,pas)` on parcourt les éléments a, $a+$ pas, $a+2$pas... Par exemple `range(10,20,2)` correspond à la liste `[10, 12, 14, 16, 18]`.

- **Imbrication de boucles.**
  Il est possible d'imbriquer des boucles, c'est-à-dire que dans le bloc d'une boucle, on utilise une nouvelle boucle.

```
for x in [10,20,30,40,50]:
    for y in [3,7]:
        print(x+y)
```

Dans ce petit programme x vaut d'abord 10, y prend la valeur 3 puis la valeur 7 (le programme affiche donc 13, puis 17). Ensuite $x=20$, et y vaut de nouveau 3 puis 7 (le programme affiche donc ensuite 23, puis 27). Au final le programme affiche :

$$13, 17, 23, 27, 33, 37, 43, 47, 53, 57.$$

Activité 4 (Boucle « pour »)**.**

Objectifs : construire des boucles simples.

1. (a) Affiche les cubes des entiers de 0 à 100.
   (b) Affiche les puissances quatrièmes des entiers de 10 à 20.
   (c) Affiche les racines carrées des entiers 0, 5, 10, 15,... jusqu'à 100.
2. Affiche les puissances de 2, de 2^1 à 2^{10}, et apprends par cœur les résultats !
3. Recherche de façon expérimentale une valeur approchée du minimum de la fonction
$$f(x) = x^3 - x^2 - \frac{1}{4}x + 1$$
sur l'intervalle $[0,1]$.
Indications.
   - Construis une boucle dans laquelle une variable i balaye les entiers de 0 à 100.
   - Définis $x = \frac{i}{100}$. Ainsi $x = 0.00$, puis $x = 0.01$, $x = 0.02\ldots$
   - Calcule $y = x^3 - x^2 - \frac{1}{4}x + 1$.
   - Affiche les valeurs à l'aide de `print("x =",x,"y =",y)`.
   - Cherche à la main pour quelle valeur de x on obtient un y le plus petit possible.
   - N'hésite pas à modifier ton programme pour augmenter la précision.
4. Cherche une valeur approchée que doit avoir le rayon R d'une boule afin que son volume soit 100.
Indications.
   - Utilise une méthode de balayage comme à la question précédente.
   - La formule du volume d'une boule est $V = \frac{4}{3}\pi R^3$.
   - Affiche les valeurs à l'aide de `print("R =",R,"V =",V)`.
   - Pour π tu peux prendre la valeur approchée 3.14 ou bien la valeur approchée `pi` du module `math`.

Activité 5 (Boucle « pour » (suite))**.**

Objectifs : construire des boucles plus compliquées.

1. Définis une variable n (par exemple $n = 20$). Calcule la somme
$$1^2 + 2^2 + 3^2 + \cdots + i^2 + \cdots + n^2.$$
2. Calcule le produit :
$$1 \times 3 \times 5 \times \cdots \times 19.$$
Indications. Commence par définir une variable `produit` initialisée à la valeur 1. Utilise `range(a,b,2)` pour obtenir un entier sur deux.
3. Affiche les tables de multiplication entre 1 et 10. Voici un exemple de ligne à afficher :
```
7 x 9 = 63
```
Utilise une commande d'affichage du style : `print(a,"x",b,"=",a*b)`.