

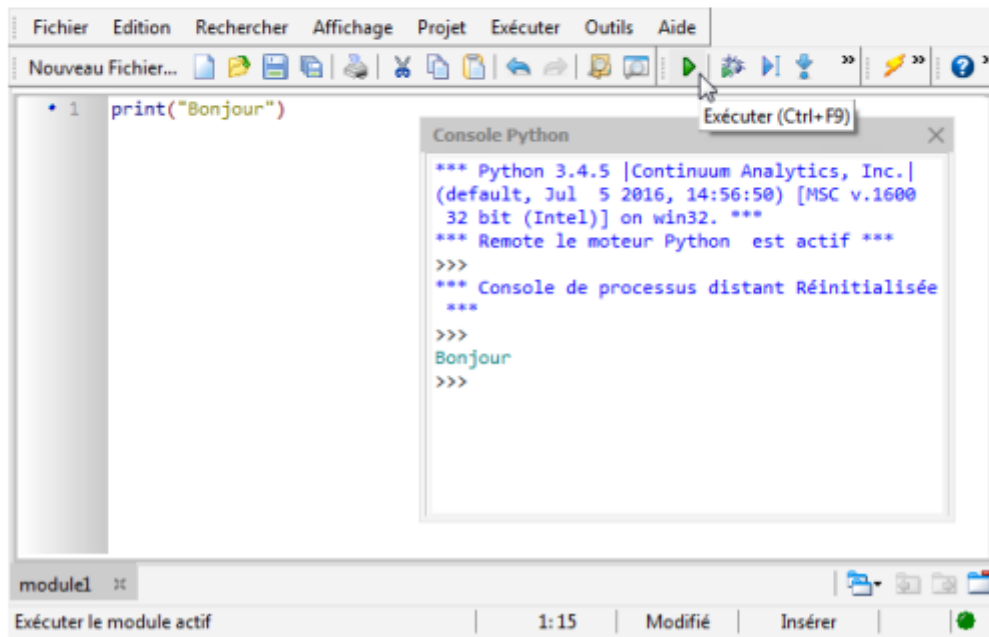
Initiation à Python

Introduction

Python est un langage de programmation inventé dans les années 90.

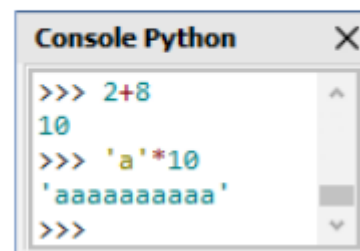
Il existe plusieurs versions, parmi lesquelles

- version en ligne : <https://repl.it/languages/python3> ;
- sur la calculatrice NumWorks ;
- EduPython (copie d'écran ci-dessous) ; Pyzo...



Un script Python est écrit dans un écran appelé **éditeur** de script, puis exécuté dans une **console** par un appui sur le triangle vert.

On peut aussi saisir des commandes directement dans la console, mais on ne peut pas les sauvegarder.



Python

I) Bilan calcul numérique :

+ - * / sont les symboles de l'addition, la soustraction, la multiplication et la division (pas de surprise)

a//b nous renvoie le quotient entier de la division euclidienne de a par b (en 20 combien de fois 3 ? 6)

a%b nous renvoie le reste de la division euclidienne de a par b ($20=3\times 6+2$)

a**b nous renvoie le nombre a exposant b : a^b .

Les priorités de calculs sont bien respectées, les calculs avec des nombres décimaux non entiers sont parfois approximatifs ($1,56\times 2,3=3,588$) mais les calculs avec des entiers sont très efficaces (2^{1000}).

Le séparateur décimal est le point . et la virgule sert à séparer deux objets (nombre, variables, ...)

II) Bilan sur les variables :

a, b, c et result sont ici **des variables**.

Le symbole =

Le **symbole =** n'est pas le même qu'en Mathématiques, c'est ici un symbole d'affectation.

a=2 peut se traduire par « affecter le nombre entier 2 à la variable a », on peut le noter naturellement $a\leftarrow 2$

a=a+1 peut donc se traduire par « affecter le nombre a+1 à la variable a », Python calcule a+1, mémorise cette somme et la nomme a.

Le symbole ==

a==b peut se traduire par « a est-il égal à b ? ». La console nous retourne alors True ou False (booléen).

De la même façon le **symbole !=** est un test :

a!=b peut se traduire par « a est-il différent de b ? ». La console nous retourne True ou False.

Le symbole *

En mathématiques, si x est un nombre, $2x = 2 \times x$.

En langage Python, on ne peut pas se passer du symbole * pour une multiplication, pour calculer $2\times a$ on doit taper $2*a$.

III) Dans l'éditeur :

Exemple 1 :

On considère la fonction f définie sur \mathbb{R} par $f(x) = 5x - 7$

a) Calculer de tête $f(-4)$

Nous allons définir la fonction f dans l'éditeur :

b) Dans l'**éditeur**, taper :

```
def f(x):  
    return 5*x-7
```

Attention, les deux points : sont d'une importance capitale. Ils montrent le début de la définition de la fonction f qui dépend de la variable x. L'indentation (décalage du bloc à droite) se fait automatiquement après ces deux points et doit être respectée.

c) Enregistrer le fichier dans vos documents sous le nom mathpython

d) Utiliser la flèche verte pour exécuter votre programme.

Vous venez alors de définir la fonction f définie par $f(x) = 5x - 7$ en langage Python.

e) Dans la **console**, taper alors :

>>>f(-4) (votre console connaît maintenant la fonction f et peut calculer l'image de -4 par f)

f) Tester alors d'autres calculs d'image par f.

Exemple 2 : On considère la fonction g définie sur \mathbb{R} par $g(x) = x^2 - 4x + 1$

Définir, dans l'éditeur, dans le même fichier mathpython, une fonction Python nommée g correspondant à la fonction définie ci-dessus.

Commentaires : penser à bien revenir à la ligne lors du début de la définition de g dans l'éditeur et vérifier les bonnes indentations (qui sont automatiques après les deux points :).

Ne pas oublier le symbole $*$ pour le « $4*x$ ».

Vérifier la bonne définition de votre fonction en testant quelques calculs d'images.

IV) Structure conditionnelle : if... else... (si alors.... sinon...)

Exemple 3 : une fonction définie par morceaux

On considère la fonction h définie sur \mathbb{R} par $h(x) = \begin{cases} x + 1 & \text{si } x \leq 1 \\ 2x & \text{si } x > 1 \end{cases}$

1) Calculer à la main les images de -5 ; -2 ; 0 ; 1 ; 3 ; 7 et 12 .

2) Dans l'éditeur taper :

```
def h(x):
    if x<=1:
        return x+1
    else:
        return 2*x
```

Puis vérifier vos calculs de la question 1.

Bilan : if ... else

La ligne « $\text{if } x \leq 1 :$ » teste si x est inférieur ou égal à 1. Si « $x \leq 1$ » est vrai (True) alors h retourne $x+1$.

La ligne « $\text{else} :$ » permet d'indiquer la consigne si x n'est pas inférieur ou égal à 1. (else : sinon)

Exemple 4 : On considère la fonction k définie sur \mathbb{R} par $k(x) = \begin{cases} x^2 & \text{si } x < 3 \\ 2x - 5 & \text{si } x \geq 3 \end{cases}$

Définir, dans l'éditeur, dans le même fichier mathpython, une fonction Python nommée k correspondant à la fonction définie ci-dessus.

V) Les modules de Python :

Certains outils du langage Python sont rangés dans des modules (bibliothèques). Lorsqu'on veut les utiliser, on doit les importer depuis ces modules. On peut importer un seul outil d'un module ou bien importer la totalité des outils de ce module.

1) Le module math

- Le nombre π n'est pas connu par la console : taper `>>>pi` dans la console et constater que Python vous signale que la variable π n'est pas définie.
- taper `>>>sqrt(2)` et constater que la fonction racine carrée n'est pas définie non plus (sqrt est l'abréviation de squared root c'est à dire racine carrée)

Taper alors `>>>from math import pi`
puis `>>>pi`

π est alors une variable que nous avons importée du module math dans laquelle est stockée une valeur approchée de π .

Par contre, nous n'avons importé ici que cet objet π du module math qui en contient bien d'autres. La fonction sqrt n'est toujours pas définie. Pour importer la totalité du module math, taper :

`>>>from math import *` (l'astérisque * signifie « tout » ici)

`>>>sqrt(2)`

Affectation et comparaison

1. Quelle valeur retourne l'algorithme suivant (à la main) ?

```
a ← 3  
b ← a + 2  
c ← a × b  
b ← c − 10  
Retourner b
```

L'affectation d'une valeur à une variable, que l'on note \leftarrow en pseudo-code, se fait par le symbole `=` en Python.

Traduire l'algorithme ci-dessus en Python puis le programmer.

```
a=3  
  
print( )
```


2. Compléter les trous ci-dessous.

```
>>> a=5
>>> a*10

>>> b=1.2
>>> b*10

>>> 2**3

>>> 10/5

>>> 10//5

>>> 11//5
```

Que permet de faire l'opérateur ****** ?
Et l'opérateur **//** ?

3. Dire si les affirmations suivantes sont vraies ou fausses.

$5 = 6$: $5 \neq 6$:
 $5 > 7$: $2 \leq 2$:

En python, les commandes pour effectuer des tests sont :

- comparaison :
- différent de :
- plus grand ; plus grand ou égal :

La réponse est vrai (True) ou faux (False)

```
>>> 5==6
False
>>> 5!=6
True
```

4. On pose $A = 600^{700}$, $B = 700^{600}$ et $C = 650^{650}$.
- a. À l'aide de Python, déterminer parmi les réels A et B lequel est le plus petit :

Recopier les commandes saisies.

```
>>> A=600**700
```

- b. Classer par ordre croissant les nombres A , B , C .

Types de variables

Il existe plusieurs types de variables en Python.

- Le type **int** désigne les entiers (2, -3, ...)
 - Le type **float** désigne les nombres à virgule ;
 - Le type **string** désigne les chaînes de caractères, *toujours écrites entre guillemets ou apostrophes* : « toto », 'bonjour', ... ;
 - Le type **bool** qui désigne les variables booléennes ne pouvant prendre que les valeurs « vrai » ou « faux » (généralement ce sont des résultats de tests, voir ci-dessus).
5. Exécuter les commandes ci-dessous et compléter les trous.

```
>>> a=5
>>> type(a)

>>> a*3

>>> b='bonjour'
>>> type(b)

>>> b*3
```

6. Donner le type des variables suivantes :

- $5-28+1$:
- $6/3$:
- `'abc'` :
- $5!=7$:

Les fonctions print et input

Exécuter le script suivant. Que se passe-t-il ?

```
a=8
b=2*a
```

Et pour celui-ci ?

```
a=8
b=2*a
print(b)
```

La fonction **print** permet aussi d'afficher plusieurs variables, à condition de les séparer par des virgules :

```
a=8
b=2*a
print("le double de",a,"vaut",b)
```

Exécuter le script ci-dessous.


```
nom=input("Comment vous appelez-  
vous ?")  
print("Bonjour",nom)
```

Que permet de faire la fonction **input** ?
.

Exécuter le script ci-dessous avec le nombre de votre
choix.

```
a=input("Choisissez un entier")  
b=2*a  
print("le double de",a,"vaut",b)
```

Le résultat est-il celui attendu ?
Donner le type de la variable **a**
Résoudre le problème rencontré à l'aide de la fonction
int() qui convertit une variable en type entier (quand
c'est possible).
Recopier le script ci-dessous.

On retiendra que la fonction **input()** renvoie une variable de type **str** et que par conséquent selon l'usage que l'on en fait, il faut la convertir en réel, avec la **float()**, ou en entier, avec **int()**.

Les fonctions en Python

Considérons la fonction f qui à un nombre associe son double. Cela s'écrit $f: x \mapsto 2x$.

En Python pour définir une fonction, on utilise la commande **def** ; pour afficher la valeur à retourner, on utilise **return**.

La définition de la fonction s'arrête lorsque l'indentation se termine.

Exécuter le script suivant.

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

Dans la console, il suffira de saisir :

```
>>> f(10)  
20  
>>> f(-8)  
-16
```

7. Pour le marché de Noël, Clara fabrique des crêpes qu'elle vend 1,5 € chacune.

- a. Créer une fonction **vente** ayant pour argument le nombre x de crêpes vendues et qui renvoie le montant total de ses ventes (la recette).

```
def vente(x):
```

Elle doit payer 58 € sa place au marché, et chaque crêpe lui coûte 0,27 € à fabriquer.

- b.** Créer une fonction **cout** ayant pour argument le nombre x de crêpes vendues et qui renvoie le coût total.

- c.** En utilisant les fonctions **vente** et **cout**, créer une fonction **benefice** ayant pour argument le nombre x de crêpes vendues et qui renvoie le montant total de son bénéfice.

Instructions conditionnelles

La résolution de certains problèmes nécessite la mise en place d'un test pour effectuer une tâche :

- si le test est positif, on effectue la tâche ;
- sinon, c'est-à-dire si le test est négatif, on effectue une autre tâche.

Le « sinon » n'est pas obligatoire. En son absence, lorsque le test est négatif, la tâche ne sera pas effectuée et l'algorithme passera à l'instruction suivante.

Le programme ci-dessous détermine, à partir de l'âge a d'une personne, si elle est mineure ou majeure.

```
Si  $a < 18$  alors
    afficher « vous êtes mineur »
Sinon
    afficher « vous êtes majeur »
```

Voici sa traduction en Python.

```
age=input("Entrez votre âge")
age=float(age)
if age < 18:
    print(" . . . . ")
else:
    print(" . . . . ")
```

Le « alors » s'écrit avec deux points en Python. L'indentation délimite une instruction conditionnelle.

8. Compléter le programme, saisir et exécuter le script avec deux âges pour s'assurer du bon fonctionnement, l'un inférieur à 18, l'autre supérieur.
9. Transformer ce script en une fonction **age** dont l'appel donnera par exemple :

```
>>> age(15)
'vous êtes mineur'

>>> age(19)
'vous êtes majeur'
```

10. On considère un algorithme qui affiche
 - « normal » si la température est strictement supérieure à 5° ;
 - « froid » si elle est comprise entre -5° et 5° ;
 - « grand froid » sinon.

On propose deux algorithmes, prenant en entrée la température T .

Compléter le second et le programmer sous forme d'une fonction **temp**.

```
Si  $T > 5$  alors  
    afficher « normal »  
Si  $(T \geq -5)$  et  $(T \leq 5)$  alors  
    afficher « froid »  
Si  $T < -5$  alors  
    afficher « grand froid »
```

```
Si . . . alors  
    afficher « normal »  
Sinon  
    Si . . . alors  
        afficher « froid »  
    Sinon  
        afficher « grand froid »
```

L'exécution donnera par exemple :

```
>>> temp(-15)  
'grand froid'
```


L'exécution donnera par exemple :

```
>>> temp(-15)
'grand froid'
```

La boucle « pour »

Il est aussi possible de demander à un ordinateur de répéter une même tâche autant de fois que l'on veut. Cela se fait grâce à ce qu'on appelle une **boucle**. Regardons de suite un exemple pour voir l'intérêt de cette notion.

L'algorithme ci-contre affiche les carrés des nombres entiers de 1 à 10.

```
Pour  $K$  de 1 à 10
afficher  $K^2$ 
```

Le principe de fonctionnement est le suivant : l'ordinateur affecte à K la valeur 1 comme il lui est demandé puis effectue les instructions comprises entre « Pour » et « Fin Pour ». Il augmente ensuite automatiquement K de 1 (donc K vaut 2) puis effectue à nouveau le bloc d'instruction. K est alors augmenté de 1 et ainsi de suite jusqu'à ce que K vaille 10.

En Python, la syntaxe est la suivante.

```
For k in range(1,11):
    print(k*k)
```

Notez qu'il faut terminer à 11 et non à 10 (l'explication est que Python comprend ceci : « pour k entier dans l'intervalle $[1; 11[$ », par conséquent si l'on écrit `range(1,10)`, la boucle s'arrêtera à $k = 9$ qui est le dernier entier de l'intervalle $[1; 10[$).

11. Saisir le script précédent et l'exécuter.

12. Que fait l'algorithme ci-contre ? Combien retourne-t-il ?


```

S ← 0
Pour K de 1 à 100
    S ← S + K
Retourner S
```

Généraliser en écrivant une fonction **somme** qui calcule la somme

$$1 + 2 + 3 + \dots + n$$

où n est donné en argument à la fonction.

```

>>> somme(4)
10
```

La boucle « tant que »

En algorithmique, on peut être amené à répéter un bloc d'instructions tant qu'une condition est vérifiée. En langage naturel, une telle répétition en boucle peut se formuler par « tant que », traduit par « while » dans la plupart des langages.

Si la condition qui suit l'instruction « tant que » est vérifiée, alors le programme exécute toutes les instructions du bloc qui suit ; à la fin de cette exécution il regarde à nouveau si la condition est vérifiée et ainsi de suite jusqu'à ce qu'elle ne le soit plus.

13. Toute boucle « pour » peut être remplacée par une boucle « tant que » à l'aide d'un compteur.

L'algorithme ci-contre affiche

lui aussi les carrés des entiers de 1 à 100.

Voici sa traduction en Python. L'expérimenter.

```
K ← 1
Tant que K ≤ 100
  afficher K2
  K ← K + 1
```

```
k=1
while k<=10:
    print(k*k)
    k=k+1
```

En revanche il existe des utilisations de « tant que » qui ne peuvent pas être remplacées par celle de « pour ».

14. L'effectif d'une population de bactéries double chaque jour. Si la population est de 100 bactéries le premier jour, au bout de combien de jours dépassera-t-elle 2 000 bactéries ? (à la main)

Si la question avait été « au bout de combien de temps la population dépassera-t-elle 6 milliards de bactéries ? », le calcul à la main aurait été fastidieux. Compléter l'algorithme ci-dessous et le programmer.

```
J ←  
P ←  
Tant que P ≤  
  P ←  
  J ←  
Retourner
```

J compte le nombre de jours et P est le nombre de bactéries le J -ième jour.

- 15.** Une balle est lâchée de 100 mètres de haut et rebondit aux $4/5^{\text{ème}}$ de sa hauteur à chaque rebond.
- Calculer la hauteur du premier et du second rebond.

- b. On considère que la balle ne rebondit plus si le rebond est inférieur à 1 cm. Au bout de combien de rebonds la balle ne rebondit plus ?

16. Chaque semaine à l'occasion de la diffusion d'une série télévisée, on retrouve 90 % des téléspectateurs de l'épisode précédent et 10 000 nouveaux. Il y avait 200 000 téléspectateurs lors du premier épisode.

Pour $n \geq 1$, on note u_n le nombre de téléspectateurs ayant regardé le $n^{\text{ième}}$ épisode.

- a. Que vaut u_1 ? Expliquer pourquoi

$$u_{n+1} = 0,9u_n + 10\,000$$

pour tout $n \geq 1$.

- b. Quel est le nombre de téléspectateurs au bout de 4 semaines ?

- c. Écrire un algorithme pour déterminer le nombre de téléspectateurs au bout d'un an (52 semaines) puis deux ans (104 semaines). Quelle conjecture peut-on faire ?

17. On considère la suite des nombres 1 ; 1 ; 2 ; 3 ; 5 ; 8 ; 13 ; 21 ; 34 ; ... dans laquelle un terme est la somme des deux précédents.

Écrire un algorithme qui demande à l'utilisateur un entier $n \geq 1$ et qui renvoie le $n^{\text{ième}}$ terme de la suite.

Par exemple pour $n = 7$, l'algorithme renverra 13.