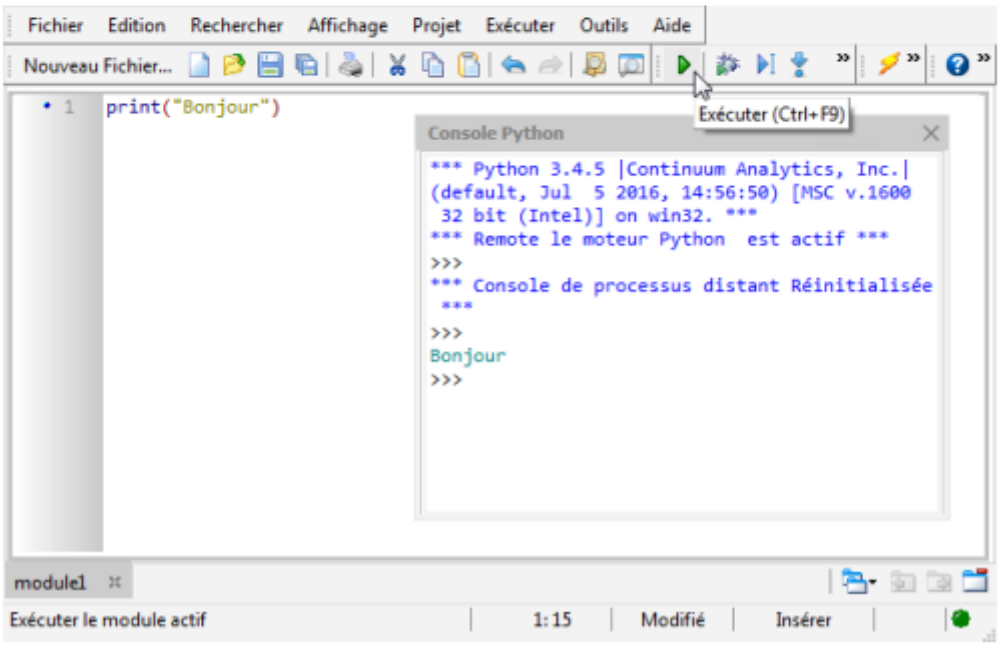
Initiation à Python

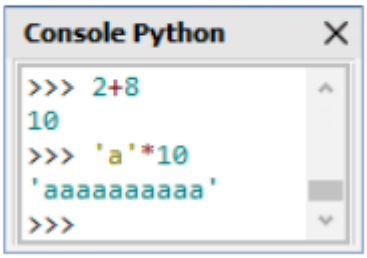
Introduction

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=3×6+**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 (1,56×2,3=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←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×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 « if x<=1 : » teste si x est inférieur ou égal à 1. Si « x<=1 » est vrai (True) alors h retourne x+1.
La ligne « 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 pi 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
pi 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 pi 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)

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 ≥ −5) et (T ≤ 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 K²
```

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

$$1 + 2 + 3 + \cdots + n$$

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 remplacée 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.

2 Les fonctions mathématiques de bases

Opérations	Interprétation	Exemples de syntaxe	Remarque
+, -, *, /	addition, soustraction, multiplication et division		
a//b	Partie entière de la division de a par b	> 12//11 1	$12 \div 11 \approx 1,0909$
a % b	Reste de la division euclidienne de a par b	> 17 % 3 2	$17 = 3 \times 5 + 2$
int(a)	partie entière	> int(12.123) 12	
divmod(a,b)	Quotient et Reste de la division euclidienne de a par b	> divmod(20,3) (6,2)	$20 = 3 \times 6 + 2$
a**b ou pow(a,b)	a Puissance b	> 2**3 ou pow(2,3) 8	$2^3 = 8$
a**(1/2)	Racine carrée $\sqrt{a}$	> 9**(1/2) 3	$\sqrt{9} = 3$
a**(1/n)	Racine $n^{\text{ième}}$ de a : $\sqrt[n]{a}$	> 27**(1/3) 3	$\sqrt[3]{27} = 3$
abs(x)	Valeur absolue de x : $\lvert x \rvert$	> abs(-5.2) 5.2	$\lvert -5.2 \rvert = 5.2$
round(a,n)	Arrondie de a à 10^{-n} près	> round(2.2563,2) 2.26	