Fiche d'aide à l'algorithmique et à la programmation (1)
Variables et affectations

- **Définition 1** : Un **algorithme** est une suite finie d'instructions et d'opérations, visant à résoudre un problème.

  Lorsqu'un algorithme est traduit dans un langage exploitable par une machine (ordinateur, calculatrice), on parle d'un **programme**, ou encore d'un **script**. Les lignes composant ce programme sont appelées du **code**.

- **Définition 2** : Une **variable** est une quantité qui peut prendre plusieurs valeurs différentes au cours de l'exécution d'un algorithme. Elle est désignée par une lettre ou un mot.

  Lorsqu'on donne à la variable V la valeur 2, on dit qu'on **affecte** la valeur 2 à la variable V.
  On le note $V \leftarrow 2$.

  Il peut être nécessaire de faire **saisir** la valeur d'une variable par l'utilisateur d'un programme, ou d'**afficher** sa valeur.

Pour réaliser une affectation	**En Python**
Affecter une ***Valeur*** *à* une ***Variable*** : $Variable \leftarrow Valeur$	`Variable=Valeur`
Quelques exemples	
Affecter la valeur 2 à la variable V : $V \leftarrow 2$	`V=2`
Affecter la valeur de la variable a à la variable b : $b \leftarrow a$	`b=a`
Calculer la valeur de $n+1$, et affecter le résultat à la variable n (ce qui revient à augmenter n de 1) : $n \leftarrow n+1$	`n=n+1`

Pour saisir la valeur d'une variable	**En Python**
Saisir la valeur de x	Si x est un **nombre réel** : `x=float(input("x = "))`
	Si x est un **nombre entier** : `x=int(input("x = "))`
	Si x est une **chaine de caractères** : `x=input("x = ")`

Pour réaliser un affichage	**En Python**
Afficher un ***Message*** suivi de la valeur d'une ***Variable***	`print("Message",Variable)`
Quelques exemples	
Afficher le message « Bonjour »	`print("Bonjour")`
Afficher la valeur de la variable A	`print(A)`
Afficher le message « $A =$ » suivi de la valeur de la variable A	`print("A = ",A)`

- **Remarque** : Il est possible d'insérer la valeur d'une variable$h$ à l'intérieur d'un message :

```
print("Il faudra rouler pendant {} heures pour accomplir le trajet".format(h))
```

Ça marche aussi avec plusieurs variables$x$ et y :

```
print("L'abscisse du point M est {} et son ordonnée est {}".format(x,y))
```

Fiche d'aide à l'algorithmique et à la programmation (2)
Fonctions

- **<u>Fonctions prédéfinies</u>** : Certaines fonctions mathématiques sont regroupées dans des **modules**. Pour pouvoir les utiliser, il faut donc importer tout ou partie du module correspondant, en début de programme.

  Pour importer la totalité du module **math** :
```
from math import *
```
  Pour importer uniquement la fonction *sqrt* du module **math** :
```
from math import sqrt
```

Description	En Python	Arguments
x^n	$x ** n$	x est un nombre réel et n est un nombre entier
Partie entière de x	$int(x)$	x est un nombre réel
Quotient de la division euclidienne de a par b	$a//b$	a et b sont deux nombres entiers
Reste de la division euclidienne de a par b	$a\%b$	
Dans le module **maths**		
$\sqrt{x}$	$sqrt(x)$	x est un nombre réel
$\cos(x), \sin(x), \tan(x)$	$\cos(x), \sin(x), \tan(x)$	x est en radians
π	pi	Constante, pas d'argument
Dans le module **random**		
Nombre entier choisi aléatoirement dans l'intervalle $[a; b]$	$randint(a, b)$	a et b sont deux nombres entiers
Nombre décimal choisi aléatoirement dans l'intervalle $[a; b]$	$uniform(a, b)$	a et b sont deux nombres décimaux
Nombre décimal choisi aléatoirement dans l'intervalle $[0; 1[$	$random()$	Fonction sans argument

sqrt : **sq**uare **r**oo**t**

Définir une fonction	En Python
$f(variable) = formule$	`def f(variable):` `  return formule`
Quelques exemples	
$f(t) = t^3 - 4\sqrt{t}$	`def f(t):` `  return t**3-4*sqrt(t)`

Si $x \leq 1, g(x) = 2x + 1$, sinon $g(x) = 3x$	`def g(x):`
	`  if x<=1:`
	`  return 2*x+1`
	`  else:`
	`  return 3*x`

$Aire(b, h) = \frac{b \times h}{2}$	`def Aire(b,h):`
	`  A=b*h/2`
	`  return A`

Fiche d'aide à l'algorithmique et à la programmation (3)
Boucles

- **<u>Définition 1</u> :** Parfois, une ou plusieurs instructions ne doivent être exécutée(s) que si une certaine condition est vérifiée. On utilise alors la **boucle conditionnelle** : **Si … Alors … Sinon**

Boucle « Si »	En Python

Si la ***Condition*** est vérifiée,
alors exécuter la ***Séquence d'instructions 1***,
sinon exécuter la ***Séquence d'instructions 2***
	`if Condition:`
	`  Séquence 1`
	`else :`
	`  Séquence 2`

- **<u>Remarque 1</u>** : La partie « Sinon … » n'est pas obligatoire.
- **<u>Remarque 2</u>** : En Python, le symbole $=$ est réservé à l'affectation. Pour écrire la condition $x = 1$, on écrira $x == 1$.

Condition	En Python
$a = b$	`a==b`
$a \neq b$	`a!=b`
$a < b$	`a<b`
$a \leq b$	`a<=b`
$a > b$	`a>b`
$a \geq b$	`a>=b`
$a = b$ **ou** $c = d$	`a==b or c==d`
$a \leq b$ **et** $c \geq d$	`a<=b and c>=d`

- **<u>Remarque 3</u>** : La fin d'une boucle est marquée par le retour au niveau d'indentation précédent.

Quelques exemples	

Si $x \leq 10$, affecter la valeur $100x$ à la variable V,
sinon affecter la valeur $750 + 25x$ à V
	`if x<=10:`
	`  V=100*x`
	`else:`
	`  V=750+25*x`

Si $n = 100$, augmenter la variable k de 1, afficher le message « Gagné ! », et afficher la valeur de k. Sinon, ne rien faire.	`if N==100:`
	`  k=k+1`
	`  print ("Gagné")`
	`  print("k = ",k)`

- **<u>Définition 2</u> :** Une **boucle bornée** est une boucle qui sera répétée un certain nombre de fois, connu à l'avance. On l'appellera aussi **« boucle Pour »,** ou **« boucle For »**.

Boucle « Pour »	En Python
Pour ***variable*** allant de ***valeur1*** à ***eur2*** , exécuter la ***séquence d'instructions***	`for variable in range(valeur1,valeur2 + 1):` `  Séquence d'instructions`

- **<u>Remarque</u>** : En Python, $\boldsymbol{range(n)}$ désigne la liste de de tous les entiers **de 0 à $\boldsymbol{n-1}$**. Cette liste comporte n nombres entiers. Par exemple : $range(10) = [0\,,1\,,2\,,3\,,4\,,5\,,6\,,7\,,8\,,9]$.

  $\boldsymbol{range(1\,,n)}$ désigne la liste de tous les entiers **de 1 à $\boldsymbol{n-1}$**. Cette liste comporte $n-1$ nombres entiers. Par exemple : $range(1\,,7) = [1\,,2\,,3\,,4\,,5\,,6]$

  Plus généralement, $\boldsymbol{range(a,b)}$ désigne la liste de tous les entiers **de $\boldsymbol{a}$ à $\boldsymbol{b-1}$**.
  Par exemple : $range(2\,,13) = [2\,,3\,,4\,,5\,,6\,,7\,,8\,,9\,,10\,,11\,,12]$

Quelques exemples	

Calculer la somme $1+2+\cdots+100$	`S=0`
	`for k in range(1,101):`
	`    S=S+k`
	`print(S)`

Simuler 1000 lancers d'un dé, et compter le nombre de fois où le 5 est sorti	`from random import *`
	
	`compteur=0`
	`for k in range(1000):`
	`    d=randint(1,6)`
	`    if d==5:`
	`        compteur=compteur+1`
	`print("Le 5 est sorti {} fois".format(compteur))`

- **<u>Définition 3</u>** : Une **boucle non bornée** est une boucle qui sera répétée tant qu'une certaine condition est vérifiée.On ne connaît donc pas à l'avance le nombre de fois où la boucle sera effectuée. On l'appellera aussi **« Boucle Tant que »**, ou **« boucle While »**.

Boucle « Tant que »	**En Python**
Tant que***Condition*** est vérifiée, exécuter la ***Séquence d'instructions***	`while Condition:` `    Séquence d'instructions`

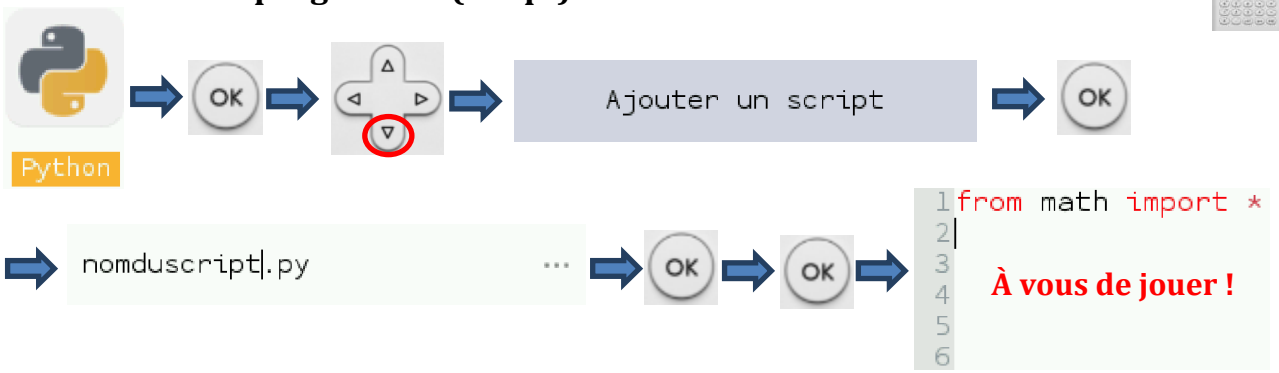
Un exemple	

Lancer un dé jusqu'à obtenir un 6, et afficher le nombre de lancers.	`from random import *`
	
	`d=0`
	`compteur=0`
	`while d!=6:`
	`    d=randint(1,7)`
	`    compteur=compteur+1`
	`print(compteur)`

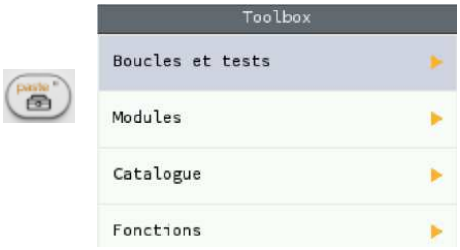
Fiche d'aide à l'algorithmique et à la programmation (4)
Écriture et exécution d'un programme en Python

Avec la calculatrice NumWorks

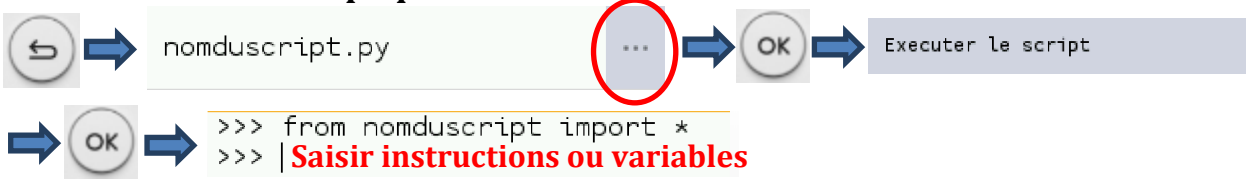
- **Pour écrire un programme (script)** :



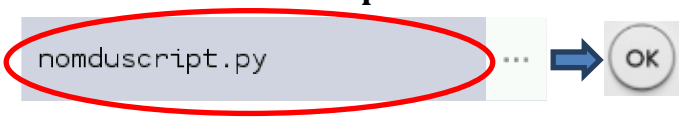
L'écriture d'un script nécessite souvent d'ouvrir la **boîte à outils :**



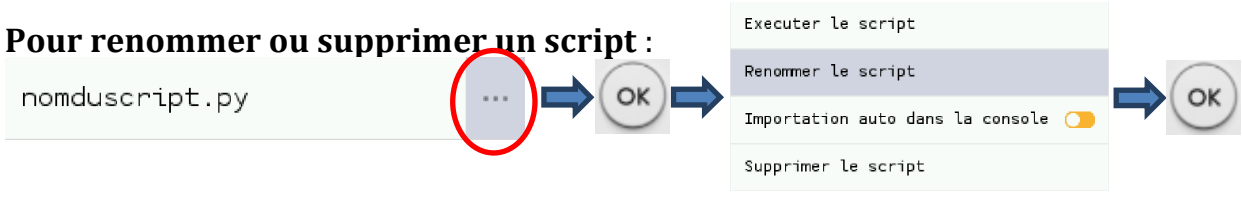
- **Pour exécuter un script que l'on vient d'écrire** :



- **Pour modifier un script** :



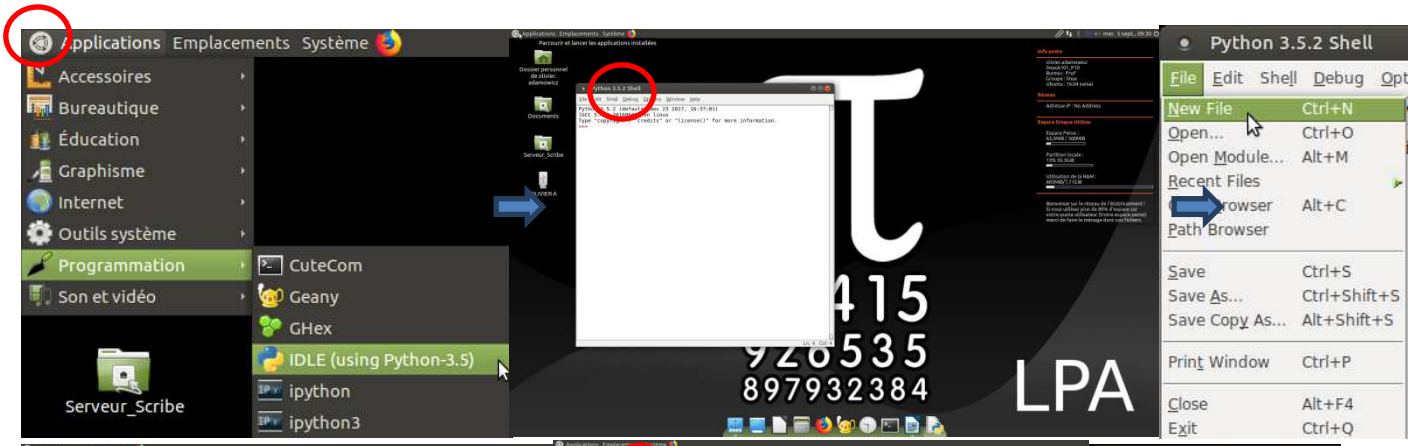
- **Pour renommer ou supprimer un script** :

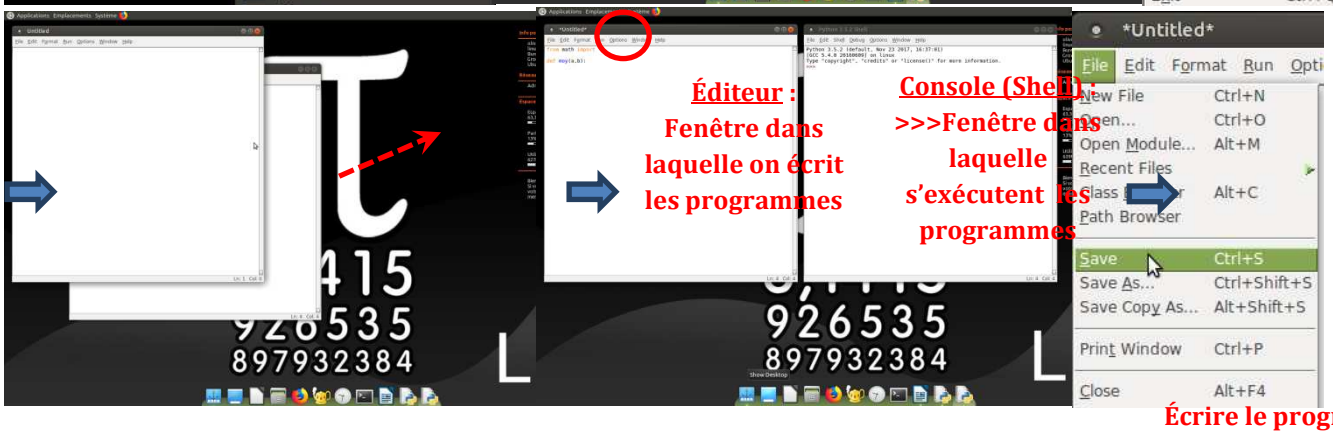


- Une fois le script écrit, testé et utilisé, pensez à **annuler l'importation automatique dans la console**.

En salle informatique, sous Linux avec IDLE

- **Pour écrire un programme :**Programmation / IDLE (using Python 3.5) / New File / Save as

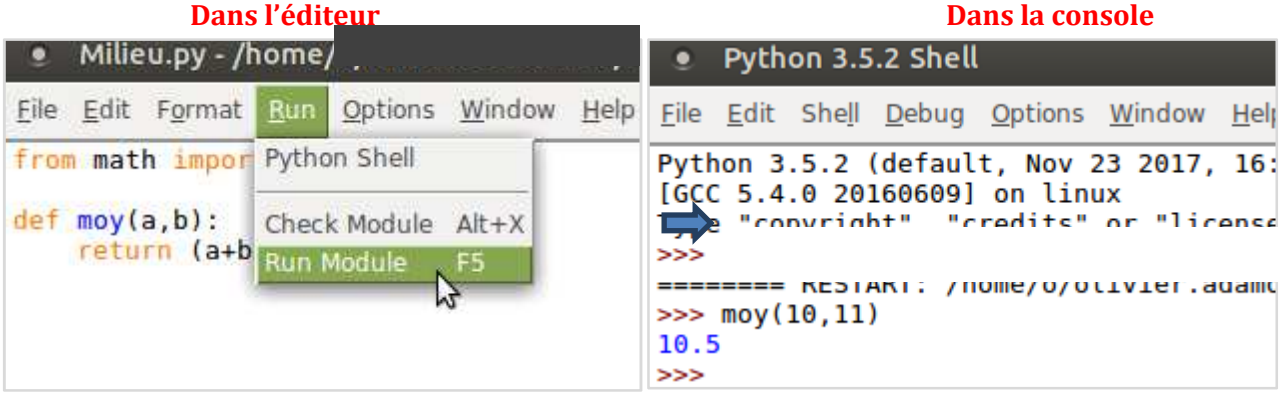




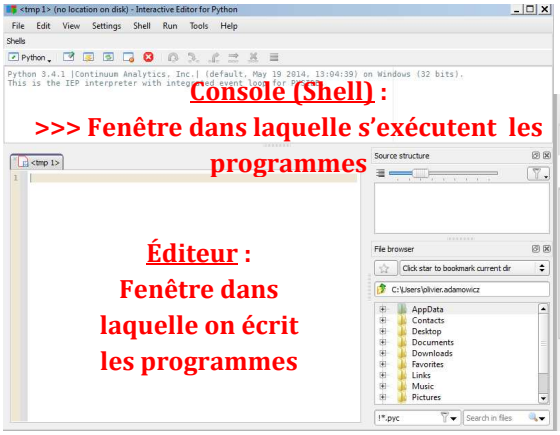
Écrire le programme dans l'éditeur, et le sauvegarder.

- **Pour exécuter un programme :**Run / Run Module

Dans l'éditeur **Dans la console**



En salle informatique, sous Windows, avec Pizo



Pour exécuter un programme : Run / Execute file