Programmation en PYTHON

Créer un nouveau programme	On saisit le programme dans l'éditeur. On va à la ligne après chaque instruction, mais on peut taper plusieurs instructions sur la même ligne en les séparant par un point-virgule.	
Importer un module	from *nom_du_module* import*	from lycee import* avec EduPython pour la bibliothèque **lycée**
Afficher A	print(A)	
Affecter à B la valeur de A	B=A	
Écrire des commentaires	On écrit les commentaires sur une ligne, précédés de #	
Tester si $A = B$ / si $A \neq B$	A==B	A !=B
Tester si $A \geq B$ / si $A \leq B$	A>=B	A<=B
A et B / A ou B	A and B	A or B
Si {condition C} Alors {instructions A} Sinon {instructions B} Fin Si	**if** {condition C}: {instructions A} **else :** {instructions B}	Il n'y a pas d'instruction de fin : c'est l'indentation (le décalage vers la droite) qui indique les instructions faisant partie de la structure conditionnelle.
Pour i variant de 1 à n {instructions} Fin Pour	**for** i in **range**(1,n+1)**:** {instructions}	• L'instruction **for i in range(n)** fait parcourir à la variable i tous les entiers de 0 à $n-1$. • L'instruction **for i in range(n_0,n)** fait parcourir à la variable i tous les entiers de n_0 à $n-1$. • L'instruction **for i in range(n_0,n,p)** fait parcourir à la variable i les entiers de n_0 jusqu'à l'entier immédiatement inférieur ou égal à $n-1$ avec un pas de p. Comme pour **if**, on utilise l'indentation pour indiquer les instructions faisant partie de la boucle.
Tant que {condition C} {instructions} Fin Tant que	**while** {condition C}**:** {instructions}	Comme pour **if** et **for**, on utilise l'indentation pour indiquer les instructions faisant partie de la boucle.
Définition d'une fonction f	**def** f(a,b,c,…)**:** y= …. **return**(y)	*On peut aussi écrire :* **return** y *au lieu de* **return**(y)
Racine carrée de x	**sqrt**(x) dans la bibliothèque **lycée** ou le module **math**	
x à la puissance n	x**n	
π	**pi** dans la bibliothèque **lycée** ou le module **math**	
Reste de la division de a par b	**a%b** avec la bibliothèque **lycée**	**fmod(a,b)** pour la calculatrice NumWorks
Nombre décimal aléatoire compris entre a et b	**uniform**(a,b) avec la bibliothèque **lycée** ou le module **random**	
Entier aléatoire compris entre a et b	**randint**(a,b) avec la bibliothèque **lycée** ou le module **random**	
Longueur d'une chaîne de caractères C / Extraire un caractère de C	**len**(C)	**C**[k] renvoie le $(k+1)^{e}$ élément de C
Concaténation de deux chaînes de caractères	"abcd" + "efg" donne "abcdefg"	
Booléens	**True** (Vrai)	**False** (Faux)
Exécuter un programme	Onglet **Exécuter**	EduPython : cliquer sur ▶

A Algorithme – Variables – Affectation

1 Algorithme et notion de variable

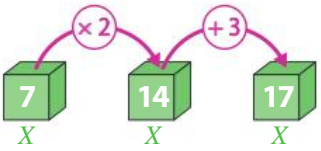
Un algorithme est une suite finie d'instructions à appliquer dans un ordre déterminé à un nombre fini de données pour arriver, en un nombre fini d'étapes, à un certain résultat.

Pour stocker un résultat, on utilise une variable. On peut se représenter une variable comme une « boîte », un emplacement de la mémoire d'un ordinateur… Pour pouvoir accéder à son contenu, on lui donne un nom.

Exemple 1 : un programme de calcul

Choisir le nombre 7
Multiplier ce nombre par 2
Ajouter 3 au résultat

On peut ici utiliser une variable, que l'on nomme X, qui contient les résultats obtenus à la fin de chaque étape du programme de calcul.



Une **variable** est désignée par un **nom**. Elle contient une **valeur**. On utilise quatre **types de valeurs** :
– **entier** (nombre entier relatif) ;
– **flottant** (nombre à virgule) ;
– **chaîne de caractères** : suite ordonnée de caractères, un caractère étant un chiffre, une lettre, un symbole… ;
– **booléen** : variable qui ne prend que deux valeurs (**Vrai** ou **Faux**), sa valeur est en général donnée par un test.

Remarques :
1) "Année" est une **chaîne de cinq caractères**. On dit que c'est une **chaîne de longueur 5**.
Le premier caractère de cette chaîne est A, son deuxième caractère est n, son troisième est n…
2) Les chaînes de caractères peuvent « s'ajouter », c'est-à-dire se mettre bout-à-bout (on parle aussi de concaténation de chaînes de caractères). Par exemple, "Année" + "2017" = "Année2017".
3) La variable b contenant le test $(6 > 4)$ est de type booléen, sa valeur est **Vrai**.

2 L'affectation

Lorsque l'on donne une valeur à une variable X, on écrit l'instruction : $X \leftarrow$ ……
On lit : « X **reçoit** …… » ou « X **prend la valeur** …… »
La nouvelle valeur remplace la valeur précédente.

Algorithme correspondant au programme de calcul de l'exemple 1 :

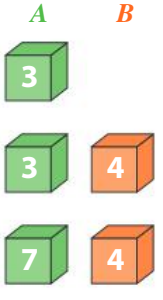
La variable X contient d'abord 7. → $X \leftarrow 7$
La valeur de X est ensuite multipliée par 2 : elle devient donc $2X$. → $X \leftarrow 2X$
Elle est enfin augmentée de 3 : elle devient donc $X + 3$. → $X \leftarrow X + 3$

Exemple 2 : un autre algorithme

Algorithme :

$A \leftarrow 3$
$B \leftarrow A + 1$
$A \leftarrow A + B$

Valeur de la variable A et valeur de la variable B après l'exécution de chaque instruction :



La valeur de A est 3 et B n'a pas encore de valeur.

Comme la valeur de A est 3, la valeur de B est : $3 + 1$, soit 4.
La valeur de A ne change pas : elle reste égale à 3.

Comme la valeur de A est 3 et que celle de B est 4, la nouvelle valeur de A est : $3 + 4$, c'est-à-dire 7.
La valeur de B ne change pas : elle reste égale à 4.

B Programmation d'un algorithme en langage Python

Les algorithmes peuvent être programmés sur un ordinateur ou une calculatrice avec un langage de programmation adapté.

Programmer un algorithme, c'est le traduire dans un langage compréhensible par un logiciel donné.

Cette page est consacrée à la découverte des premières notions du langage de programmation Python.

1 La programmation en Python

Quand on ouvre l'interface d'une distribution Python, on découvre deux fenêtres : une fenêtre d'édition des programmes, **l'éditeur**, et une fenêtre où le programme s'exécute, la **console**.

Dans la console, on peut :
– faire des calculs, définir des variables et les intégrer dans des calculs ;
– écrire et exécuter un programme ;
– exécuter un programme saisi dans l'éditeur et demander les valeurs prises par les variables de ce programme.

```
# Créé par Michel, le
from lycee import *
x=3
y=5
z=x+y
x=z**3
y=y-z/4
```

Console Python

```
*** Python 3.4.5 |Continuum Analytics,
 (default, Jul  5 2016, 14:56:50) [MSC
.1600 32 bit (Intel)] on win32. ***
>>> x
512
>>> y
3.0
```

2 Les commandes de base

a) Créer un nouveau programme : dans Fichier, choisir Nouveau.
On peut aussi choisir directement Nouveau Fichier *dans certaines versions.*
b) Instruction d'affectation : c ← a s'écrit en code Python : c = a.
c) Pour séparer deux instructions successives d'un programme, on peut soit aller à la ligne ; soit séparer ces deux instructions par un point-virgule (;).
d) Pour exécuter un programme, cliquer sur Exécuter (sur ▶ ou sur F5 selon les versions).
Remarque : on peut écrire un commentaire dans un programme en le précédant du symbole #. Ce texte n'est pas pris en compte lors de l'exécution du programme.
e) Pour tester l'égalité de deux valeurs, on utilise l'opérateur ==.
Exemple : on saisit dans la console c = 3, puis c==5. On affecte à la variable c la valeur 3, puis on teste si celle – ci est égale à 5. C'est faux, donc le résultat affiché est : False.
Remarque : l'instruction print(a) permet d'afficher la valeur de la variable a dans un programme.

3 Les commandes de calcul

Certaines commandes nécessitent un module **math** pour être obtenues. On l'obtient avec la commande **from math import*** ou, avec EduPython, en choisissant le modèle **Lycée** après Nouveau Fichier.
• Les symboles opératoires +, –, ×, / s'écrivent respectivement en Python : +, -, *, /.
• x à la puissance n s'écrit : x**n.
• Le reste de la division de a par b s'écrit : a%b (ou fmod(a,b) avec la calculatrice NumWorks).
Avec le module **math** :
• La racine carrée de x ($x \geq 0$) s'écrit : sqrt(x) et le nombre π s'écrit : pi.

4 Chaînes de caractères et booléens

• Les chaînes de caractères se définissent à l'aide de guillemets doubles ou simples.
Par exemple : "papa" ou 'papa'.

Obtenir la longueur d'une chaîne de caractères C	**len(C)**
Extraire un caractère d'une chaîne de caractères C	**C[k] renvoie le (k+1)e élément de C**
Concaténer deux chaînes de caractères	**"abcd" + "efg" donne "abcdefg"**

• Le booléen **Vrai** est : True. Le booléen **Faux** est : False.

C Les fonctions

On peut simplifier l'écriture des programmes en utilisant des **fonctions**, sur le modèle des fonctions numériques étudiées en mathématiques.

1 Exemple de fonctions

Exemple 1 : calcul d'une distance d'arrêt

La distance d'arrêt d (en mètres) d'un véhicule roulant à une vitesse v (exprimée en km.h^{-1}) sur une route sèche peut être évaluée par la formule : $d = 0{,}005v^2 + 0{,}278v$.
La distance d dépend de la variable v. Pour programmer ce calcul, on va ainsi définir une fonction, comme en mathématiques : on lui donne un nom, **arrêt** et on dit qu'elle a pour **argument** v.
En langage Python, on définit cette fonction par la commande **def** suivi du nom de la fonction, puis de son argument v, suivi de deux points.

deux points

```
def arrêt(v):
    d=0.005*v**2+0.278*v
    return(d)
```

indentation

Les autres instructions sont **indentées** par rapport à la première ligne.
Après le calcul de d, on renvoie le résultat par la commande **return**.
Pour calculer la distance d'arrêt d'un véhicule roulant à 90 km.h^{-1}, on saisit **arrêt** (90) dans la console : on trouve environ 65,5 m.

Exemple 2 : calcul de la surface de la peau

La surface S en mètres carrés de la peau d'un adulte est donnée approximativement par la formule : $S = \frac{\sqrt{L \times M}}{6}$, où L est la taille de la personne exprimée en mètres et M sa masse exprimée en kilogrammes.
Pour programmer ce calcul, on définit une fonction appelée **surf** qui a ici deux **arguments** L et M et qui retourne S, celle-ci sera écrite **surf**(L,M) dans le programme.

```
def surf(L,M):
    return(sqrt(L*M)/6)
```

Pour calculer la surface corporelle d'un adulte de 1,80 m et 75 kg, on saisit **surf**(1.8,75) dans la console : on trouve environ 1,94 m^2.

2 Définition – Programmation

Une fonction est un bloc d'instructions qui a reçu un nom et dont le fonctionnement dépend d'un certain nombre de paramètres (les **arguments** de la fonction). La fonction renvoie un résultat (au moyen de la commande **return**) ; le programme s'arrête après **return**.

Programmation d'une fonction en langage Python

```
def nom_fonction(liste des arguments):
    bloc d'instructions
    return(résultat)
```

• La programmation d'une fonction commence toujours par **def**, suivi du nom que l'on donne à la fonction, suivi des arguments de la fonction : **cette ligne se termine par deux points** (:).
• Les deux points marquent le démarrage du bloc d'instructions définissant la fonction : **toutes ces instructions sont indentées**, c'est-à-dire décalées vers la droite par rapport à la première ligne.
On ajoute en tête de chaque ligne du bloc le même nombre d'espaces.
• Pour renvoyer la valeur de la variable d, on peut écrire : **return**(d) ou **return** d.
• On peut utiliser une fonction dans la console en donnant des valeurs aux arguments (dans le bon ordre).

Propriétés

• Une fonction ne renvoie qu'un seul résultat (ce peut être une liste de résultats).
• Une fonction peut n'avoir aucun argument.
Par exemple, la fonction **essai**() ci-contre peut être utilisée dans un programme où on résout des équations.

```
def essai():
    return("il y a une unique solution")
```

• Une fonction peut être appelée dans un autre programme : il suffit pour cela de l'insérer dans une instruction en saisissant son nom et les valeurs des arguments.

D Instruction conditionnelle (If)

1 Structure d'une instruction conditionnelle

Exemple : développement de photos

Un site internet de développement de photos propose le tirage sur papier des photos au tarif de 0,11 € l'unité ; le tarif passe à 0,08 € l'unité pour une commande d'au moins 200 photos.
On veut créer un algorithme donnant le montant dépensé pour un nombre n de tirages.
Pour cela, on doit introduire une instruction conditionnelle dans l'algorithme :
– si le nombre de photos n est strictement inférieur à 200, le montant dépensé est $n \times 0{,}11$, puisque le prix d'une photo est alors 0,11 € ;
– si le nombre de photos n est supérieur ou égal à 200, le montant dépensé est $n \times 0{,}08$, puisque le prix d'une photo est alors 0,08 €.

On définit deux variables : l'une n représente le nombre de tirages, l'autre M le montant dépensé

```
Si n < 200                    ← Début de l'instruction conditionnelle : si n est inférieur à 200
| Alors M ← 0,11 × n          ← alors le montant dépensé M est 0,11 × n
| Sinon M ← 0,08 × n          ← sinon le montant dépensé M est 0,08 × n
Fin Si                        ← Fin de l'instruction conditionnelle
```

C'est la structure alternative **« Si… Alors… Sinon… »** qui permet d'écrire **l'instruction conditionnelle** dans l'algorithme : un test est effectué sur une condition C, et le résultat du test décide de l'exécution de la phase de traitement.

```
Si {condition C}
| Alors {instructions A}
| Sinon {instructions B}
Fin Si
```

Si la condition C est vérifiée, seules les instructions A sont exécutées.
Si la condition C n'est pas vérifiée, seules les instructions B sont exécutées.

Remarque : on peut aussi utiliser la structure incomplète : **« Si… Alors… »** ; dans ce cas, si la condition C n'est pas vérifiée, l'exécution de l'algorithme continue après le **Fin Si**.

2 Programmation

Programmation d'une instruction conditionnelle :

	Python	Remarques
Si {condition C} Alors {instructions A} Sinon {instructions B} Fin Si	**if** {condition C} : {instructions A} **else** : {instructions B}	En langage Python, il n'y a pas d'instruction pour indiquer la fin de l'instruction conditionnelle : c'est l'indentation qui décale vers la droite les instructions A et B.

Programme relatif à l'algorithme vu en exemple :

```
def photo(n):
    if n<200:
        M=0.11*n
    else:
        M=0.08*n
    return(M)
```

- Dans le programme ci-contre, une première indentation indique les instructions faisant partie de la fonction photo, et dans le programme de cette fonction, une seconde indentation indique les instructions A et B de l'instruction conditionnelle.
- L'instruction **return**(M) permet de renvoyer le contenu de la variable M.
- Le "else" est aligné avec le "if" qui lui correspond.
- Le fait de taper ":" puis "Entrée" pour passer à la ligne provoque l'indentation automatique dans l'éditeur.

E Boucle bornée (For)

1 Structure de la boucle « POUR »

On peut répéter un bloc d'instructions un certain nombre de fois fixé au départ : on utilise alors la boucle « POUR ».

Exemple : croissance d'un village

Un village compte aujourd'hui 2 300 habitants. Le village étant en pleine croissance, sa population augmente chaque année de 150 habitants.

On souhaite élaborer un algorithme donnant le nombre d'habitants de ce village dans n années.

Pour cela, on définit une variable P qu'on initialise à 2 300, puis on répète n fois l'opération qui consiste à ajouter 150 à P : le nombre de répétitions (ou « itérations ») est *connu au départ* puisque c'est le nombre n d'années.

On parle de « calcul itératif ».

```
P ← 2300
Pour k variant de 1 à n
|P ← P + 150
Fin Pour
```

- La variable P est initialisée à 2 300.
- Début de la boucle : k prend la valeur 1…
- On ajoute 150 à P pour avoir la nouvelle valeur de P.
- Fin de la boucle : si k est inférieur à n, k augmente de 1, sinon on sort de la boucle.

On suppose que $n = 3$.

	k	P	
Avant le début de la boucle		2 300	← Au début, la variable P est initialisée à 2 300.
Fin de la 1re itération	1	2 450	← $k = 1$ et on ajoute 150 à P donc $P = 2\,300 + 150 = 2\,450$.
Fin de la 2e itération	2	2 600	← $k = 2$ et on ajoute 150 à P donc $P = 2\,450 + 150 = 2\,600$.
Fin de la 3e itération	3	2 750	← $k = 3$ et on ajoute 150 à P donc $P = 2\,600 + 150 = 2\,750$.

La variable k contrôle le nombre de répétitions. Dans l'exemple ci-dessus, k prend pour valeur initiale 1 et pour valeur finale n en augmentant de 1 à chaque fois (cette augmentation constante est appelée « pas »).

2 Programmation

Programmation de la boucle bornée :

	Python	Remarques
Pour k variant de d à n |{instructions} **Fin Pour**	**for** k **in range**(d, $n + 1$): {instructions}	L'instruction **for** k **in range**(d, $n + 1$) fait parcourir à la variable k tous les entiers de d à n. Lorsque $d = 0$, on peut remplacer **range**(d, $n + 1$) par **range**($n + 1$).

En langage Python, les deux points « : » marquent le début du bloc d'instructions de la boucle **for**. Il n'y a pas d'instruction de fin de boucle **for** : c'est l'indentation (le décalage vers la droite) qui indique les instructions faisant partie de la boucle.

Programme relatif à l'algorithme vu en exemple :

On peut programmer cet algorithme en langage Python à l'aide d'une fonction :

```
def population(n):
    P=2300
    for k in range(1,n+1):
        P=P+150
    return(P)
```

- Dans ce programme, une première indentation indique les instructions faisant partie de la fonction population, et dans le programme de cette fonction, une seconde indentation indique l'unique instruction faisant partie de la boucle **for**.
- L'instruction **return**(P) permet de renvoyer le contenu de la variable P : ainsi lorsque l'on saisit population(3) dans la console, la valeur 2 750 est renvoyée.

F Boucle non bornée (While)

1 Structure de la boucle « TANT QUE »

On peut répéter un bloc d'instructions tant qu'une condition reste vérifiée : on utilise alors la boucle TANT QUE.

Exemple : les rebonds d'une balle

Une balle lâchée d'une hauteur de 90 centimètres rebondit chaque fois qu'elle touche le sol aux $\frac{3}{5}^{e}$ de la hauteur du rebond précédent.

On souhaite élaborer un algorithme donnant le nombre de rebonds au bout duquel la hauteur atteinte par la balle est strictement inférieure à 1 centimètre.

On définit la variable H égale à la hauteur du rebond exprimée en centimètres. On doit répéter plusieurs fois l'instruction « H prend la valeur $\frac{3}{5} \times H$ », sans savoir à l'avance le nombre de ces répétitions.

Pour cela, on teste si la variable H est supérieure ou égale à 1 en début de boucle et le traitement dans la boucle est réalisé tant que cette condition est vérifiée.

Pour compter le nombre de rebonds effectués par la balle, on introduit un « compteur » R : il est initialisé à 0, et , chaque fois que la boucle est parcourue, sa valeur augmente d'une unité.

$H \leftarrow 90$	La variable H est initialisée à 90, qui est la hauteur initiale.
$R \leftarrow 0$	Le nombre R de rebonds est initialisé à 0.
Tant que $H \geqslant 1$	On entre dans la boucle si la hauteur H est supérieure ou égale à 1 cm.
$H \leftarrow \frac{3}{5} \times H$	À chaque rebond, la hauteur est multipliée par $\frac{3}{5}$.
$R \leftarrow R + 1$	Il y a un rebond de plus, donc R augmente d'une unité.
Fin Tant que	On revient au début de la boucle pour tester si la hauteur H est ou non supérieure à 1 cm.

C'est la boucle « **Tant que…** » qui permet de répéter ce calcul.
Cette boucle est utilisée lorsque l'on veut recommencer un même bloc d'instructions jusqu'à valider une condition de sortie donnée à l'avance.
Avec une hauteur initiale de 90 cm, les valeurs des variables R et H (en centimètres) sont données ci-contre.
C'est la structure itérative avec fin de boucle conditionnelle.

Étapes	H	R	Condition vérifiée
Avant le début de la boucle	90	0	Oui
1er passage dans la boucle	54	1	Oui
2e passage dans la boucle	32,4	2	Oui
…	…	…	…

2 Programmation

Programmation de la boucle non bornée :

	Python	Remarques
Tant que {condition C} {instructions} **Fin Tant que**	**while** {condition C} : {instructions}	En langage Python, il n'y a pas d'instruction de fin de boucle **while** : c'est l'indentation (le décalage vers la droite) qui indique les instructions faisant partie de la boucle.

Programme relatif à l'algorithme vu en exemple :
On peut programmer cet algorithme en langage Python à l'aide d'une fonction :

```
def rebonds():
    H=90
    R=0
    while H>=1:
        H=H*3/5
        R=R+1
    return(R)
```

• Dans le programme ci-contre, une première indentation indique les instructions faisant partie de la fonction rebonds, et dans le programme de cette fonction, une seconde indentation indique les instructions faisant partie de la boucle **while**.

• L'instruction **return**(R) permet de renvoyer le contenu de la variable R : ainsi, lorsque l'on saisit rebonds() dans la console, la valeur 9 est renvoyée.

Programmation en PYTHON avec les calculatrices

Pour créer un programme

CASIO

❶ Sélectionner le menu Python à l'aide des flèches du curseur, puis valider avec EXE.
Sélectionner NEW (F3).

❷ On complète au-dessous de « Nom du script » en donnant un nom au programme, puis on valide avec EXE.
On se trouve dans l'éditeur, prêt à saisir les instructions.

NUMWORKS

❶ Sélectionner l'icône Python à l'aide des flèches du curseur, puis valider avec OK.

❷ Sélectionner Ajouter un script avec les flèches du curseur, puis OK.

❸ Donner un nom au programme (suivi de .py déjà noté), puis deux fois sur OK.
On se trouve alors dans l'éditeur.

TEXAS INSTRUMENTS

❶ Sélectionner le menu **apps** (touche résol), puis le sous-menu **PyAdaptr**.
Valider avec entrer.

❷ Sélectionner **Nouv** (touche zoom).

❸ Donner un nom au programme, suivi de **OK** (touche graphe).
On se trouve alors dans l'éditeur.

Pour saisir un programme

CASIO

On peut écrire les instructions lettre à lettre ou les chercher dans CATALOG (SHIFT 4).
Dans le catalogue, on obtient les instructions en tapant la première lettre de leur nom. On accède aux catégories par CAT (F6) :
2 :Built-in : def :return, for, if, while, and, or…
3 :math : math import…
4 :random : random import…
On a la barre d'outils suivante :

FILE | RUN | SYMBOL | CHAR | A⇔a

❶ En sélectionnant SYMBOL (F3), on a accès aux symboles suivants :

: | ; | # | ' | "

❷ Le menu CHAR (F4) donne accès aux lettres, aux chiffres...

❸ Par appui sur F6 ▷, on accède aux onglets suivants :
COMMAND | OPERAT | JUMP | SEARCH

❹ En sélectionnant COMMAND (F6), on a accès à if, for et while.

❺ En sélectionnant OPERAT (F2), on a accès aux fonctionnalités :

= | ! | > | < | %

La touche EXIT permet de sortir d'un menu.

NUMWORKS

Utiliser les menus déroulants accessibles avec la touche Toolbox.
On passe d'un menu à l'autre avec les flèches du curseur.
Pour ouvrir le menu, on le sélectionne et on valide par OK.

❶ **Le menu Boucles et tests** comprend quatre sous-menus : **For**, **If**, **While** et **Conditions** (comme x<y, x==y…).

❷ **Le menu Modules** contient plusieurs modules (bibliothèques de fonctions) que l'on peut importer.
Par exemple : **math**, **cmath** ou **random**. La liste des fonctions d'un module s'obtient en sélectionnant le module.

❸ **Le menu Catalogue** comprend des fonctions comme **min**, **max**, **sin**…

❹ **Le menu Fonctions** comprend les instructions **def function(x):** et **return**.

La touche ⮌ permet de revenir au menu précédent.

TEXAS INSTRUMENTS

Dans l'éditeur, on a accès à différents menus.

❶ **Menu Fns** (touche f(x)), avec plusieurs onglets, parmi eux :
- l'onglet **Ctl** permet de sélectionner toutes les instructions if, for, while…
- l'onglet **Ops** comprend les tests (x==y, x>y…), les connecteurs or, and et les booléens True, False.
- l'onglet **Fonc** donne accès à def fonction() et return.
- l'onglet **Modul** pour les modules math et random.

❷ **Menu a A #** (touche fenêtre) : lettres de l'alphabet, certains symboles (#, ", …), des connecteurs (and, or) et des booléens (True, False).

❸ **Menu Outils** (touche zoom) : il permet d'insérer une indentation, une ligne…

❹ **Menu Script** (touche graphe) : il permet d'accéder à tous les programmes de la calculatrice.

On sort d'un menu avec la commande **Echap** (touche f(x)).

Pour exécuter un programme

CASIO

Le programme étant saisi, on choisit RUN (F2), puis on tape le nom de la fonction pour exécuter le programme dans la console.

NUMWORKS

On se place dans la console.
Pour cela, on choisit Console d'execution, puis OK.

TEXAS INSTRUMENTS

Dans le gestionnaire de scripts, choisir à l'aide des touches de défilement le programme voulu, puis sélectionner le menu **Exéc**.