# Instructions du langage Python

## Affectations et opérations (mettre des valeurs en mémoire)

| Instruction | Action effectuée |
|---|---|
| N = 4 | L'ordinateur met le nombre 4 dans la case mémoire N : 4→N |
| N = N+1 | L'ordinateur ajoute 1 à la case N puis met le résultat obtenu dans la case N, à la place du nombre qui s'y trouvait |
| A = 5**3 | L'ordinateur met le résultat de $5^3$ dans la case A 125→A<br>Remarque : On obtient $\sqrt{5}$ avec 5**0.5 car $\sqrt{5} = 5^{1/2}$ |
| M = "Voilà !" | L'ordinateur met le texte (appelé **chaîne de caractères**) entre guillemets (sans les guillemets) dans la case M Voilà !→M |

## Sorties (faire écrire sur l'écran)

| Instruction | Action effectuée | Affichage écran |
|---|---|---|
| **print**(1+2*3/4) | L'ordinateur écrit sur l'écran le résultat du calcul | 2.5 |
| **print**("Bonjour !") | L'ordinateur écrit sur l'écran le texte (appelé **chaîne de caractères**) entre guillemets (sans les guillemets) | Bonjour ! |
| N = 7<br>**print**("A = ", N) | L'ordinateur écrit sur une même ligne le texte entre guillemets puis le contenu de la case mémoire N | A = 7 |

## Autres opérations mathématiques

Pour utiliser cette bibliothèque, il faut écrire au début du programme : **from math import ***

| Instruction | Action effectuée |
|---|---|
| **from math import *** | L'ordinateur charge dans sa mémoire toutes les instructions mathématiques de la bibliothèque **math** |
| X = **sqrt**(5) | Racine carrée $\sqrt{5}$ 1,732…→X |
| K = **exp**(2) | Exponentielle $e^2$ 7,389…→K |
| T = **log**(3) | Logarithme népérien $\ln 3$ 1,098…→T |

# Tirages aléatoires

Pour utiliser cette bibliothèque, il faut écrire au début du programme : **from random import ***

| Instruction | Action effectuée |
|---|---|
| **from random import *** | L'ordinateur charge dans sa mémoire toutes les instructions de la bibliothèque **random** |
| X = **randint**(1, 50) | L'ordinateur crée un nombre **entier** aléatoire entre 1 et 50 compris, puis le met dans la case X |
| A = **random**() | L'ordinateur crée un nombre **décimal** aléatoire dans l'intervalle $[0;1[$ puis le met dans la case A |

# Les tests If ... Else...

| Instruction | Action effectuée |
|---|---|
| **if** [condition] **:**<br>[instructions]<br><br>[suite du programme] | Si la [condition] est vraie, l'ordinateur exécute toutes les [instructions] qui sont en alinéa[1].<br>Si la [condition] est fausse, l'ordinateur n'exécute pas les [instructions].<br><br>Une fois le « if » terminé, l'ordinateur continue et exécute dans tous les cas la [suite du programme] |

| ***Exemples de conditions*** | | | |
|---|---|---|---|
| if x **==**1 : | Si la case x contient le nombre 1… | if x **>** 1 : | Si x contient un nombre supérieur à 1… |
| if x **!=**1 : | Si x contient un nombre différent de 1… | if x**>=**1 : | Si x est supérieur ou égal à 1… |

| Instruction | Action effectuée |
|---|---|
| **if** [condition] **:**<br>[…<br>instructions 1<br>…]<br>**else :**<br>[…<br>instructions 2<br>…]<br><br>[suite du programme] | Si la [condition] est vraie, l'ordinateur exécute toutes les [instructions 1] qui sont en alinéa, puis passe à la [suite du programme].<br><br>Sinon (la [condition] étant fausse), l'ordinateur n'exécute pas les [instructions 1] mais exécute les [instructions 2], puis passe à la [suite du programme]. |

# Les listes

Une liste s'utilise pour stocker en mémoire un « tableau » de valeurs.

| Instruction | Action effectuée |
|---|---|
| L = [3,8,-9] | L'ordinateur met les nombres 3 ; 8 et –1 en mémoire dans une liste L : $(3;8;-9)\rightarrow L$ |
| A = L[0] + L[2] | Vu que L[0]=3 et L[2]=–9 on obtient alors : $-6\rightarrow A$ |
| L.append(18) | L'ordinateur ajoute 18 à la fin de la liste L : $(3;8;-1;18)\rightarrow N$ |

[1] une tabulation supplémentaire par rapport à l'alignement à gauche du mot **for** qui marque le début de la boucle.

# La boucle FOR

Elle s'utilise lorsqu'on veut faire exécuter plusieurs fois les mêmes instructions, et qu'on peut indiquer à l'ordinateur le nombre de répétitions désiré.

| Instruction | Action effectuée |
|---|---|
| **for** k **in** range(n) **:**<br>[…<br>instructions<br>…]<br><br>[suite du programme] | 1) L'ordinateur met 0 dans la case k et exécute les [instructions] qui sont en alinéa[2].<br>2) L'ordinateur met 1 dans la case k et exécute à nouveau les [instructions] qui sont en alinéa.<br>3) L'ordinateur met 2 dans la case k et exécute à nouveau les [instructions] qui sont en alinéa.<br>… etc. …<br>n) Enfin, l'ordinateur met n–1 dans la case k et exécute à nouveau les [instructions] qui sont en alinéa.<br>Il passe ensuite à la [suite du programme] qui n'est plus en alinéa.<br>Par exemple pour « k in range(10) », la 1re valeur de k sera 1 et la dernière sera 9. |
| **for** k **in** range(p,n) **:**<br>[…<br>instructions<br>…] | Même chose à part que la première valeur mise dans la case k est la valeur de p.<br>Par exemple pour « k in range(1,7) », la 1re valeur de k sera 1 et la dernière sera 6. |

# La boucle WHILE

Elle s'utilise lorsqu'on veut faire exécuter plusieurs fois les mêmes instructions : ces instructions seront répétées tant qu'une **condition** sera **vraie**. La boucle s'arrêtera donc dès que la condition sera fausse.

| Instruction | Action effectuée |
|---|---|
| **while** [condition] **:**<br>[…<br>instructions<br>…]<br><br>[suite du programme] | L'ordinateur teste la [condition].<br>**Si la [condition] est vraie**, il exécute les [instructions] qui sont en alinéa.<br>L'ordinateur teste la [condition] à nouveau.<br>Si la [condition] est encore vraie, il exécute à nouveau les [instructions] qui sont en alinéa.<br>Et ainsi de suite, tant que la [condition] est vraie.<br>Si celle-ci est toujours vraie, il continue ainsi à l'infini et ne passe jamais à la suite du programme.<br><br>**Si la [condition] devient fausse**, l'ordinateur passe à la [suite du programme] qui n'est plus en alinéa. |

# Les fonctions

| Instruction | Action effectuée |
|---|---|
| **def** Nom_fonction(var1, var2…) **:**<br>[…<br>instructions<br>…]<br>**return** [résultat]<br><br>[suite du programme…<br>avec appel de la fonction **Nom_fonction :**<br>a = Nom_fonction(3,2.1,…)] | L'ordinateur… n'exécute rien !<br><br>Mais il définit une **fonction** (aussi appelée un « **sous-programme** » ou une « **procédure** ») : c'est un « mini-programme » qui ne s'exécutera que s'il est « appelé » dans la suite du programme.<br><br>*(voir exemple)* |

[2] une tabulation supplémentaire par rapport à l'alignement à gauche du mot **for** qui marque le début de la boucle.