

**compétences
attendues**

2^{de}

SNT

Sciences numériques et technologie

2^e édition

**TRAVAILLER
EN AUTONOMIE**

*Les connaissances
du programme*

*Les capacités
et compétences*

*Les exercices
avec tous les corrigés*



Cours

Tout le monde a déjà eu dans les mains un itinéraire, une notice de montage, une recette de cuisine. La notion qui rassemble ces exemples est la notion d'algorithme.

1 Algorithme

Histoire

Le terme « **algorithme** » provient d'une traduction latine du nom du mathématicien arabe Abu Ja'far Muhammad ibn **Musa al-Khwarizmi** qui a participé à la traduction de nombreux manuscrits scientifiques grecs.



Définition

Un **algorithme** est une séquence ordonnée d'instructions exécutées de façon logique, mais non intelligente.

Remarque

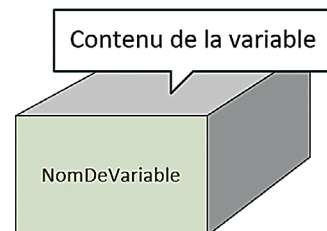
« Logique et non intelligente » signifie que l'exécutant (homme ou machine) doit strictement appliquer les instructions dans l'ordre, les unes après les autres, sans réfléchir à la justesse du résultat ni à la manière dont il est obtenu.

2 Variables et types de données

Variables

Pour fonctionner, un algorithme a besoin de mémoriser des données (saisies par l'utilisateur, résultats à afficher, résultats intermédiaires lors d'un traitement...) afin de pouvoir les réutiliser ensuite. Pour cela il utilise ce qu'on appelle une **variable**.

On peut considérer qu'une variable est une sorte de petit tiroir dans lequel on range une donnée. Ce tiroir est identifiable par une petite étiquette (un nom)



afin de pouvoir le retrouver parmi les très nombreux autres tiroirs qui constituent la mémoire.

Remarque

De façon générale, pour le nom des variables, on se limitera aux caractères alphanumériques non accentués et au tiret de soulignement (*underscore*). Toujours dans un souci de bonne lisibilité du programme, il est important de choisir des noms reflétant le rôle joué par la variable en question.

Types de données

Les variables peuvent être de plusieurs types possibles. Les plus fréquents en Python sont :

Type	Pour qui ?	Exemples
int	Les entiers	..., -5, -1, 0, 2, 6, ...
float	Les réels	-5.28, 13.0, 18.7
str	Les chaînes de caractères (texte)	'Janvier', '28b6', '13'
bool	Les booléens : uniquement deux valeurs possibles	True (Vrai) ou False (Faux)
list	C'est une liste ou tableau dans lequel on peut stocker plusieurs valeurs idéalement de même type	[0, 1, 2, 4], ['Janvier', 'Février', 'Mars']

Pour visualiser le type d'une variable *x* en python, il suffit d'utiliser la fonction **type(x)** dans la console. Il existe également de nombreux IDE, comme Spyder, permettant d'accéder directement aux types des variables. On verra que dans certains cas, il sera nécessaire de passer d'un type à un autre.

3 Affectation de variables

Définition

L'affectation est le fait d'attribuer du contenu à une variable.

D'une façon générale, avant leur utilisation, les variables doivent toujours être initialisées, soit par une saisie de l'utilisateur, soit par l'affectation d'une valeur fixée. Dans le langage naturel, l'affectation est symbolisée par le symbole ← selon la syntaxe suivante : *NomDeVariable* ← *Valeur*. En Python, l'affectation est définie avec le signe =.

Exemples

```

Longueur = 6.3
Nom = "Tartampion"
Prenom = 'Roberto'
Nb_elements = 15
Test = False
lst_mois = [
    "Janvier",
    "Février",
    "Mars",
    "Avril",
    "Mai",
    "Juin",
    "Juillet",
    "Août",
    "Septembre",
    "Octobre",
    "Novembre",
    "Décembre"
]

```

Remarque

L'affectation d'une chaîne de caractères s'établit entre guillemets doubles ou simples.

L'explorateur de variables de Spyder permet de lister les variables utilisées, leur type et leur contenu :

Nom	Type	Taille	Valeur
Longueur	float	1	6.3
Nb_elements	int	1	15
Nom	str	1	Tartampion
Prenom	str	1	Roberto
Test	bool	1	False
lst_mois	list	12	['Janvier', 'Février', 'Mars', 'Avril', 'Mai', 'Juin', 'Juillet', 'Aoû ...

4 Interaction avec l'utilisateur**La saisie**

La plupart des scripts nécessitent à un moment donné une intervention de l'utilisateur. Dans un script en mode texte, on utilise la fonction intégrée **input()**. L'utilisateur

est invité à entrer des caractères au clavier, la saisie se termine lorsque l'utilisateur appuie sur la touche *Entrée*. Cette fonction fournit en retour une **chaîne de caractères** correspondant à ce que l'utilisateur a saisi. Cette chaîne peut alors être assignée à une variable quelconque, convertie, etc.

```
prenom = input("Veuillez saisir votre prénom : ")
```

À l'intérieur des parenthèses, on donne un petit message explicatif destiné à l'utilisateur.

Remarque

Cette fonction ne retourne que des chaînes de caractères. Si l'on souhaite saisir un entier ou un réel, il faudra convertir le type *String* en *Int* ou *float*.

L'affichage

Nous souhaitons souvent afficher des informations. La fonction permettant de le faire en python est la fonction **print()**. Elle permet d'afficher les valeurs données en argument (c'est-à-dire entre parenthèses). Par défaut, ces valeurs seront séparées les unes des autres par un espace, et le tout se terminera par un saut à la ligne.

```
print("Je m'appelle" , prenom)
```

5 Les opérateurs

On utilise des **opérateurs** (symboles ou mots réservés) pour manipuler les variables et leurs contenus. Lorsqu'il y a plus d'un opérateur dans une expression, l'ordre dans lequel les opérations doivent être effectuées dépend des règles de priorité. Sous Python, les règles de priorité sont les mêmes que celles qui vous ont été enseignées en cours de mathématique. Un opérateur fonctionne avec des opérandes et est lié à un type donné.

Les principaux opérateurs arithmétiques (sur des nombres 'int' ou 'float')

$x + y$	Addition
$x - y$	Soustraction
$x * y$	Multiplication
x / y	Division décimale de x par y
$x // y$	Quotient de la division euclidienne de x par y
$x \% y$	Reste de la division euclidienne de x par y (Modulo)
$x ** y$	Puissance : x^y

Les principaux opérateurs sur des chaînes de caractères

$ch1 + ch2$	Concaténation (les chaînes $ch1$ et $ch2$ sont mises bout à bout)
$len(ch1)$	Nombre de caractères de la chaîne $ch1$
$ch1[i]$	$(i + 1)^{ème}$ caractère de la chaîne $ch1$

Les opérateurs de comparaison (pour former des expressions booléennes)

$x < y$	Tester si x est inférieur à y
$x \leq y$	Tester si x est inférieur ou égal à y
$x > y$	Tester si x est supérieur à y
$x \geq y$	Tester si x est supérieur ou égal à y
$x == y$	Tester si x est égal de y
$x != y$	Tester si x est différent de y

Remarque

Il ne faut pas confondre le test d'égalité avec l'affectation !

Les principaux opérateurs sur les listes

$len(L)$	Donner le nombre d'éléments de la liste L
$L.append(e)$	Ajouter l'élément e à la fin de la liste L
$del L[i]$	Supprimer le $(i + 1)^{ème}$ élément de la liste L
$L.reverse()$	Inverser les éléments de la liste L (le premier élément sera le dernier, etc.)
$L.sort()$	Trier la liste L dans l'ordre croissant

Les opérateurs logiques (pour réunir des expressions booléennes)

$x \text{ and } y$	Intersection
$x \text{ or } y$	Union
$\text{not } y$	Négation

Exercices

Compétence attendue

- Savoir identifier le type des variables.

Exercice 1.1

Décrire et expliquer un programme

- Que vaut la variable `ch3` après l'exécution du script suivant ?

```
ch1 = "az"
ch2 = "er"
ch3 = ch1 + ch2 + "ty"
```

Exercice 1.2

Décrire et expliquer un programme

- Que vaut la variable `a` après l'exécution du script suivant ?

```
a = 0.10
b = 3
c = 100
a = a + b/c
```

Exercice 1.3

Décrire et expliquer un programme

- Que vaut la variable `tot` après l'exécution du script suivant ?

```
ch1 = "az"
ch2 = "er"
ch3 = ch1 + ch2 + "ty"
```

**Compétence
attendue**

- Savoir répondre à un problème par la réalisation d'un programme informatique.

Exercice 1.4

Formaliser un problème
Concevoir une solution en réponse à un problème
Faire preuve d'autonomie

- Écrire un script permettant l'affichage de l'aire d'un rectangle en ayant demandé la saisie de sa longueur et de sa largeur.

Exercice 1.5

Formaliser un problème
Concevoir une solution en réponse à un problème
Faire preuve d'autonomie

- Écrire un script demandant à l'utilisateur un nombre entier constitué de 3 chiffres, puis affichant la somme des chiffres de ce nombre.

Exercice-bilan

Exercice-bilan 1.1

🕒 5 min

► Qu'affichent les scripts suivants ? (A, B, C ou D ?)

Scripts	A	B	C	D
1. <pre>a = 12 b = 8 c = a a = b b = c print(a,b)</pre>	8 8	8 12	12 8	12 12
2. <pre>nombre1 = 13.4 Nombre1 = 10.5 Nombre1 = Nombre1 - 5.5 print(nombre1)</pre>	5.0	10.5	13.4	7.9
3. <pre>nb1 = '10' nb2 = '2' tot = nb1 + nb2 print(tot)</pre>	12	102	210	Le script génère une erreur.
<pre>produit_str = '7*9' produit_int = int(produit_str) print(produit_int)</pre>	7*9	63	produit_int	Le script génère une erreur.
5. <pre>Chaine = 'Bonjour' print(Chaine[3])</pre>	j	n	3	Chaine[3]