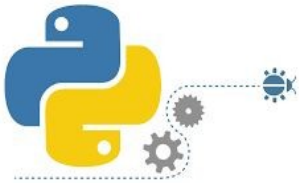
Algorithmique et Programmation en Python

Table des matières



L'objectif de la séquence 1 est de mettre en place les premières notions de programmation : variables, affectations et structure conditionnelle en langage Python en abordant assez vite l'écriture d'un programme à l'aide de fonctions.

Fiche élève n°1 :Variables et affectations en Python

Voici une affectation : A ← 2.
Le nom de la variable A désigne un espace de stockage c'est à dire un emplacement précis dans la mémoire vive de l'ordinateur.
A cet emplacement est stocké une valeur bien déterminée : ici 2. Le terme valeur ne signifie pas nécessairement un nombre mais peut être un caractère, une chaîne de caractère, une liste …

En Python, l'opération d'affectation est représentée par le signe = .

1°) Compléter le tableau ci-dessous avec les valeurs de la variable X lors du déroulement de l'algorithme :

	X
X ← 0	0
A ← 1	
B ← 9	
C ← 4	
D ← 8	
X ← X*10 + B	
X ← X*10 + C	
X ← X*10 + D	

Traduction de cet algorithme en langage Python :

```
# affectations
X = 0
A = 1
B = 9
C = 4
D = 8
X = A
X = X*10 + B
X = X*10 + C
X = X*10 + D
```

Le symbole # sert à placer un commentaire dans le script du programme.

Les valeurs affectées aux différentes variables sont des entiers. On dit qu'elles sont de **type** entier.

En Python, il n'est pas nécessaire de définir le type des variables avant de pouvoir les utiliser. Le typage se fera automatiquement : si la valeur fournie est un nombre entier alors le type de la variable dans laquelle on aura affecté cette valeur sera « entier ».

2°) Entrez ce programme dans l'éditeur Python (IDLE) en insérant des instructions d'affichage du contenu de la variable X à chaque étape.

Affichage du résultat : print « message », nom de la variable.
Cliquer sur File/New Window pour ouvrir une fenêtre dans l'éditeur Python (IDLE) qui contiendra le script du programme.
Pour exécuter votre programme, utiliser la commande Run/ run module (F5). Le programme s'exécute dans une nouvelle fenêtre (Shell).

3°) Que produit l'affichage de l'algorithme ci-dessous :

```
Entrer a , b
a ← b
b ← a
Afficher a, b
```

Ecrire un algorithme qui échange le contenu de deux variables a et b et le programmer en langage Python.

4°) Ecrire un programme en langage Python qui permute les valeurs stockées dans les variables a, b et c (permutation circulaire).

Algorithmique et Programmation en Python

Fiche élève n°2 : Type « entier », type « flottant »

1°) Ecrire un algorithme qui prend en entrées deux points A et B du plan donnés par leurs coordonnées et qui renvoie en sortie les coordonnées du milieu du segment [AB].

2°) Traduire cet algorithme en langage Python puis le tester dans les cas suivants :

a) A(1 ; 5) et B(-3 ; 3)	b) A(-5 ; 0) et B(4 ; 1)	c) A(2,2 ; -1) et B(-1 ; 0,6)

On obtient :

a)	b)	c)
entrez l'abscisse du point A : 1 entrez l'ordonnee du point A : 5 entrez l'abscisse du point B : -3 entrez l'ordonnee du point B : 3 les coordonnees du milieu sont (-1 ; 4)	entrez l'abscisse du point A : -5 entrez l'ordonnee du point A : 0 entrez l'abscisse du point B : 4 entrez l'ordonnee du point B : 1 les coordonnees du milieu sont (-1 ; 0)	entrez l'abscisse du point A : 2.2 entrez l'ordonnee du point A : -1 entrez l'abscisse du point B : -1 entrez l'ordonnee du point B : 0.6 les coordonnees du milieu sont (0.6 ; -0.2)

3°) Que produit le calcul à la main dans chaque cas ? Que pensez-vous de ces résultats ?

Ajouter les instructions suivantes en fin de programme :

```
print type (abscisse_milieu)
print type (ordonnee_mileu)
```

Quel est le type des variables recevant les coordonnées du milieu dans chaque cas ?

4°) Le type **float** (nombre en virgule flottante).

La division a / b en Python version 2.6 ou 2.7 produit le quotient entier des deux nombres a et b si ceux-ci sont entiers et le quotient décimal approché si a et/ou b sont flottants.
Pour corriger le programme, nous allons utiliser la fonction *float()* qui permet de transformer le type d'une variable contenant un nombre en flottant.

La saisie des coordonnées se fera de la façon suivante :
abscisse_A = float(input(« entrez l'abscisse du point A : »)

Refaire les tests ci-dessus, conclure.

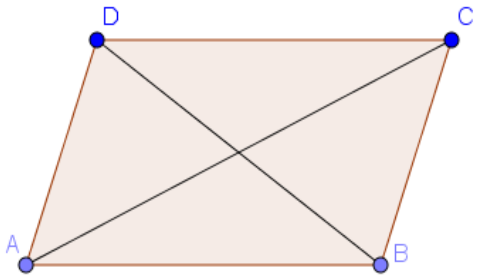
5°) Ecrire un programme en langage Python qui affecte les valeurs 3, 5 et 7 à trois variables a, b et c et qui affiche le résultat de l'opération a – b/c.
Le résultat est-il correct numériquement ?
Si ce n'est pas le cas, modifier votre programme pour qu'il le soit.

6°) Devinez ce qu'affichera le programme ci-contre :

```
a= 17
b = a/2
c = float (b)
d = float(a)/2
print b
print type(b)
print c
print type(c)
print d
print type(d)
```

Fiche élève n°3 : Instruction conditionnelle

On souhaite écrire un programme permettant de tester si quatre points donnés par leurs coordonnées forment un parallélogramme. Le test s'appuiera sur la propriété des milieux des diagonales du quadrilatère formé par ces quatre points.



Nous allons reprendre le programme de calcul des coordonnées d'un milieu et le modifier pour qu'il réponde au problème posé.

1°) Modifier le programme « ***milieu.py*** » pour qu'il prenne en entrées quatre couples de coordonnées correspondant aux quatre points A, B, C et D et qu'il renvoie les coordonnées des milieux des diagonales du quadrilatère ABCD.

2°) Structure conditionnelle en Python .

```
if condition :
        traitement 1
# veiller à l'indentation des instructions d'un même bloc
else :
        traitement 2
```

Remarques:
Un **si** n'est pas nécessairement suivi d'un **sinon.**
Il existe également elif (sinon si) suivi d'une condition : qui s'utilise à la place de else lorsque les conditions sont imbriquées : contraction de else if.

Modifier votre programme en utilisant une condition sur les milieux des diagonales, afin qu'il affiche le message attendu dans chaque cas.

En langage Python, les principaux symboles pour les comparaisons s'écrivent :

x est égal à y	**x = = y**
x est strictement plus grand que y	**x > y**
x est strictement plus petit que y	**x < y**
x est plus grand ou égal à y	**x >= y**
x est plus petit ou égal à y	**x <= y**
x est différent de y	**x != y**

3°) Recherche de maximum et premier tri:

1°) Ecrire un algorithme lisant les valeurs de deux nombres a et b et affectant le maximum de a et b à la variable *maxi* et le minimum à la variable *mini*.

2°) Ecrire un algorithme lisant les valeurs de deux nombres a et b et affectant le maximum de a et b à la variable a et le minimum à la variable b.

3°) Ecrire un algorithme lisant les valeurs de trois nombres a, b et c et affectant le maximum de a, b et c à la variable *maxi*.
Programmer cet algorithme en langage Python.

4°) Ecrire un programme en langage Python qui prend trois nombres a, b et c en entrée et qui affiche ces nombres dans l'ordre croissant.

Fiche élève n°4 : Utiliser des fonctions en programmation

Pour écrire le programme « parallélogramme », il a fallu reprendre le programme « milieu ».
Les fonctions en programmation permettent de décomposer un programme complexe en une série de sous-programmes plus simples.
Voici comment écrire la fonction « milieu » :

```
def milieu(x,y,z,t):
    abscisse_milieu = (x + z)/2
    ordonnee_milieu = (y + t)/2
    return abscisse_milieu,ordonnee_milieu
```

La syntaxe en Python est la suivante :

```
def nomdelafonction ( arguments) :   # ligne d'en-tête qui se termine par :
        instruction 1
        instruction 2                # bloc d'instructions
        instruction 3
        ……..
        return resultat              # return signifie renvoie et s'utilise seulement si la
                                     fonction produit un résultat.

# arguments est une information qu'il faudra fournir à la fonction  mais les parenthèses peuvent rester
vides si la fonction ne nécessite aucun argument.
```

Dans le cas de la fonction « milieu », il faut fournir quatre données qui sont les coordonnées des deux points.

Et voici comment on peut écrire le programme « *milieu.py* » à l'aide de cette fonction :

```
# coordonnees d'un milieu avec une fonction
def milieu(x,y,z,t):
    abscisse_milieu = (x + z)/2
    ordonnee_milieu = (y + t)/2
    return abscisse_milieu,ordonnee_milieu

abs_A = float(input("entrez l'abscisse du point A : "))
ord_A = float(input ("entrez l'ordonnee du point A : "))
abs_B = float(input("entrez l'abscisse du point B : "))
ord_B = float(input ("entrez l'ordonnee du point B : "))

print "les coordonnees du milieu sont" ,
print milieu(abs_A,ord_A,abs_B,ord_B)
```

La fonction « *milieu* » est appelée en dernière ligne, on lui donne alors les quatre paramètres effectifs abs_A, ord_A, abs_B, ord_B qui prendront la place des paramètres formels x, y, z, t prévus au moment de l'écriture de la fonction.

1°) Réécrire le programme « parallélogramme » en utilisant la fonction « milieu ».

2°) Ecrire une fonction « distance » qui calculera la distance entre deux points du plan.

3°) Ecrire un programme utilisant les fonctions précédentes qui prend en entrées quatre points A, B, C, D et qui renvoie la nature du quadrilatère ABCD.
Tester votre programme avec les points suivant :

A(8 ; 5)	B(4 ; 2)	C(4 ; -3)	D(8 ; 0)	losange
A(-2 ; 2)	B(3 ; 1)	C(-1 ; -3)	D(-6 ; -2)	parallélogramme
A(1 ; -2)	B(2 ; 1)	C(-4 ; 3)	D(-5 ; 0)	rectangle
A(4 ; 1)	B(2 ; 5)	C(-2 ; 3)	D(0 ; -1)	carré
A(-6 ; 1)	B(3 ; -5)	C(9 ; 4)	D(0 ; 10)	carré
A(-1 ; -2)	B(3 ; 0)	C(0 ; 1)	D(-4 ; -1)	parallélogramme
A(2 ; 5)	B(-1 ; 4)	C(-2 ; -3)	D(-5 ; -3)	quelconque