Lycée Ibn Taymia
CPGE
Marrakech

Chapitre 1 Algorithmique de base et programmation python

Mr. EL AAKIF MOHAMED

Cours de première année

1. Les variables

1.1 Définition :

une variable est une donnée (**zone mémoire**) dont la valeur est susceptible d'être changée durant le déroulement de l'algorithme.

1.2 Nom d'une variable :

Appelé aussi **identificateur**, il est formé d'une suite de **lettres**, de **chiffres** et de **traits de soulignement** '_' , dont le **premier caractère est obligatoirement une lettre**.

1.3 Types des variables en algorithmique et en langage python :

Le type d'une variable caractérise le type des valeurs qu'elle peut prendre.

En algorithmique	En python
entier	**int**
réel	**float**
caractère	**-**
Chaîne de caractères	**str**
Logique ou booléen (vrai, faux)	**bool** (True, False)
-	**complex**

1.4 Déclaration des variables :

Exemple de déclaration des variables en algorithmique :

```
Variable
        Nbr_disques : ENTIER;
        Rayon, Perimetre, Surface : REEL;
```

Remarque : **Python a un typage dynamique (il n'offre pas de déclaration explicite).**

2. Affectation :

C'est l'action qui permet d'attribuer à **une variable**, une valeur numérique ou bien une valeur résultante de l'évaluation d'une expression arithmétique ou logique, cette valeur doit être **compatible** avec le type de la variable destinataire.

2.1 Syntaxe en algorithmique :

Nom_Variable ← Expression ;

***Exemples* :**

- A ← 3 ; (A doit être de type ENTIER ou REEL).
- B ← 4*A+3,12 ; (B doit être de type REEL).
- Test ← A<B ; (Test doit être de type LOGIQUE)
- Lettre ← 'Z' ; (Lettre doit être de type CARACTERE).

2.2 Syntaxe en python:

Nom_Variable = Expression ;

Exemples :

- C = ‘4’ ; (C est de type str).
- B = 3 + 2j ; (B est de type complex).
- x = 4 ; y = 6 ; (x et y sont de type int).
- Z = x + y*1j ; (Z est de type complex).
- A = complex(x , y) ; (Z est de type complex, équivalent à x + y*1j)
- S1 = ‘informatique’ ; (S1 est de type str).
- S2 = "informatique" ; (S2 est de type str).

Affectation parallèle:

Le langage Python permet d’affecter plusieurs variables en n’utilisant qu’une seule fois l’opérateur d’affectation.

Exemple:

x , y , z = 2 , 5 , 7

L’indentation en Python (Tabulation) : Dans un même bloc, deux instructions de même profondeur logique doivent avoir strictement la même indentation.

3. Les fonctions d'entrée / sortie :

3.1 – Fonction d'entrée (Fonction de lecture) :

C'est l'instruction qui permet à l'utilisateur de fournir à l'algorithme les valeurs des données variables.

A- Syntaxe en algorithmique :

LIRE(V1,V2,…,Vn) ;

où V1,V2,…,Vn sont des variables mais pas forcément de même type.

B- Syntaxe en python:

L'expression input(*message*) affiche message à l'écran, attend que l'utilisateur valide par "Entrée" une réponse au clavier, et renvoie cette réponse sous forme de chaîne de caractères.

Exemple :

```
var1 = input('entrez une chaîne:') # ou str(input()) pour lire une chaîne de caractères.
var2 = int(input('entrez un entier: '))  # pour lire un entier.
var3 = float(input('entrez un réel: '))  # pour lire un réel.
var4 = complex(input('entrez un nombre complexe: '))  # pour lire un complexe.
```

3.2 – Fonction de sortie (Fonction d'écriture) :

C'est l'instruction qui permet à l'algorithme d'afficher pour son utilisateur des messages ou des résultats de calculs.

A- Syntaxe en algorithmique :

ECRIRE(VAL1, VAL2,…, VALn) ;

Où VAL1, VAL2,…, VALn est une suite d'objets (constantes, variables, expressions ou constantes chaîne de caractères).

***Exemple* : ECRIRE("Bonjour") ;**

B- Syntaxe en python:

La fonction print( ) permet d'afficher des informations à l'écran.

Exemple :

```
a = 5
print("Le contenu de a est:", a)
```

Exécution : Le contenu de a est: 5

Remarque: print("message",end=' ') # l'expression end=' ' empêche le retour à la ligne

3.3 – Fonctions prédéfinies :

3.3.1 – Fonctions prédéfinies en algorithmique :

Quotient(N1, N2) ; { Retourne le quotient de la division euclidienne entière de N1 par N2 }

Reste(N1, N2) ; { Retourne le reste de la division euclidienne entière de N1 par N2 }

Abs(X) ; { retourne la valeur absolue d'un entier ou d'un réel }

Racine(X) ; {retourne la racine carrée d'un entier ou d'un réel }

Rang(caractère) ; { retourne son code ASCII }

Exemple : Rang('A') ; { retourne la valeur 65 }

Car(code) ; { retourne le caractère correspond à ce code ASCII }

Exemple : Car(65) ; { retourne le caractère 'A' }

3.3.2 – Fonctions prédéfinies en python :

abs(x); **valeur absolue, module d'un nombre complexe.**

pow(x, y) ; calcule x^y. Par défaut $0^0 = 1$

A- Les fonctions prédéfinies du module math

Pour utiliser les fonctions mathématiques usuelles, on importe le module math de Python via l'instruction suivante en début du programme :

from math import *

Fonctions du module math:

e ; pi : constantes e = 2.718281828459045 et π = 3.141592653589793

log(x); log2(x); log10(x); log(x,b) : logarithme népérien, de base 2, de base 10, de base b

Fonctions trigonométriques : cos(x); sin(x); tan(x); acos(x); asin(x); atan(x).

Fonctions hyperboliques : cosh(x) sinh(x) tanh(x); acosh(x); asinh(x); atanh(x).

floor(x); ceil(x); trunc(x) : partie entière [x], entier plafond, tronque vers l'entier.

sqrt(x); fabs(x) : calcule respectivement $\sqrt{x}$, |x|.

factorial(x); gamma(x) : factorielle x! (x dans N), fonction d'Euler Γ(x).

B- Les fonctions prédéfinies du module cmath

Pour utiliser les fonctions prédefinies du module cmath, on écrit l'instruction suivante au début du programme:

from cmath import *

Fonctions du module cmath:

exp(z);

phase(z) ; # argument de z, exprimé en radians dans l'intervalle]- π ; π]

polar(z) ; # forme polaire de z ; équivalent à (abs(z), phase(z))

rect(r , θ) ; # passage de la forme polaire $re^{i\theta}$ à la forme cartésienne x + iy

4. les expressions arithmétiques et logiques

4.1- Syntaxe en algorithmique :

A - les expressions arithmétiques :

Une expression peut être composée d'opérateurs, d'opérandes et de parenthèses '**(**' et '**)**'.

Les opérations arithmétiques usuels sont : **+**, **-**, *****, /, **%**

(a%b) est le reste de la division entière de a par b.

B - les opérateurs de comparaison (relationnels) :

Les opérateurs de comparaison sont : $<$, $\leq$, $=$, $\neq$, $\geq$, $>$, ils sont utilisables pour les opérandes de type : entier, réel ou caractère et retournent un booléen

C - les opérateurs logiques :

Les opérateurs logiques sont : et, ou, non. Ils s'appliquent à des opérandes booléens.

A	Non(A)
faux	vrai
vrai	faux

A	B	A ET B	A OU B
faux	faux	faux	faux
faux	vrai	faux	vrai
vrai	faux	faux	vrai
vrai	vrai	vrai	vrai

4.2 Les opérateurs en python:

A- Les opérateurs arithmétiques :

Les opérateurs arithmétiques usuels : + , - , * , /
x ** y calcule x à la puissance y. Par défaut $0^0 = 1$
x // y ; x % y quotient et reste dans une division euclidienne entre entiers

B- Les opérateurs de comparaison :

< ; <= ; > ; >=
== (égal à)
! = (différent de)

C- Les opérateurs logique:

or (ou logique)
and (et logique)
not (négation logique)

5. Les commentaires

Les commentaires ne sont pas pris en compte lors de l'exécution.

5.1- Commentaires en algorithmique :

{ ceci est un commentaire }

5.2- Commentaires en python :

 # ceci est un commentaire