# LES STRUCTURES & LES TYPES DE DONNEES

## I. L'objet « variable » :

### 1) Définition :

Une variable est un espace mémoire de l'ordinateur dans lequel une ***valeur*** est ***stockée*** et qui peut subir des modifications au cours d'un traitement. Elle est caractérisée par un nom ou (**identificateur**), un type et un contenu.

### 2) Syntaxe :

*Nom_Variable = valeur_variable*

*Exp* :

```
a = 3   # On définit la variable « a » avec sa valeur 3
b = 5   # Une autre variable b initialisée à 5
```

a → 3

Remarque : le contenu d'une variable peut être modifié soit par une opération d'affectation soit par une opération de lecture ( a = input ( ' donner a : ' ))

En Python, une variable :

- ✓ doit commencer par une lettre ou un Underscore « _ » (*Exp :* age _age )
- ✓ contient des chiffres
- ✓ ne contient pas de caractères spéciaux (@ - ; ? ! …)
- ✓ ne contient pas d'espaces
- ✓ peut avoir des underscores pour les noms de variables combinés (*Exp :* age_personne)

***Attention*** ! *Python distingue les majuscules des minuscules (Python est sensible à la casse)*

*Donc age, AGE ou Age sont des variables différentes.*

*Exp :*

x1 est un identificateur valide
_x1 est un identificateur valide
1x_y n'est pas un identificateur valide car il commence par un chiffre
x-y n'est pas un identificateur valide car il contient le caractère -

*Constations :*

*Le modèle de données de Python est basé sur les objets.*

*Toute donnée manipulée est un objet avec un identifiant, un type et une valeur.*

- *id() : renvoie un entier représentant l'identifiant interne d'un objet*
- *type() : renvoie le type d'un objet.*
- *dir() : liste l'ensemble des fonctionnalités d'un objet.*

**Activité :**

a=32 et b=2.5 Déterminer le type, l'identifiant et les fonctionnalités des objets a et b.

```
a=32
print (type(a))   # ……
print (id(a))     #……
print (dir(a))    # ……
b=2.5
print (type(b))   #……
print (id(b))     # ……
print (dir(b))
```

## II. Les types de données :

En Python, les principaux types standards sont :

- Les entiers numériques : les entiers
- Les nombres décimaux (flottants) que nous appellerons réels
- Les chaînes de caractères
- Le booléen

Il existe d'autres types (tels que ; le type liste, tuple, complexe ,……)

## 1) Le type entier (Int) :

### a) Les opérateurs arithmétiques sur les entiers :

A part les opérateurs arithmétiques usuels (+ - * / ), on ajoutera deux opérateurs : **Div( // ) et Mod(%)**

| Opérateur | Syntaxe | | Exemple : |
|---|---|---|---|
| | Algorithme | Python | |
| Addition | + | + | 2+2 = ..... |
| Soustraction | - | - | 2-2 = ..... |
| Multiplication | * | * | 3 * 2 = ..... |
| Division | / | / | 5/2 = ..... |
| Puissance | Puissance | ** | 3**2 = ..... |
| Division entière | // | // | 5//2 = ..... |
| Reste de la division entière | % | % | 5%2 = ..... |

*Tels que ;*
**// :** Donne le quotient (q) de la division entière.
**% :** (modulo) : donne le reste (r) de la division entière.

| A | B |
|---|---|
| r | q |

**q** = A // B
**r** = A **%** B

### b) Ordre de priorité des opérateurs :

L'ordre de priorité est le suivant :

1. Les parenthèses ()
2. La multiplication *, la division /, la division entière // et le reste de la division entière %
3. L'addition + et la soustraction –

N.B. Si les opérateurs ont la même priorité, on commence de gauche vers la droite.

***Applications :***

Evaluez les expressions arithmétiques suivantes :

| Instruction algorithmique | Traduction en Python | Evaluation : |
|---|---|---|
| A ← ((40 DIV 3) MOD 4 ) + 6 | | |
| B ← 22 - 25 MOD (2 * 3) + 1 | | |
| C ← 15 – (12 MOD 2) * 3 - 16 | | |
| D ← (18 + 20 MOD 3) * 2 - 18 | | |
| E ← carré (2) - 25 DIV 7 * 10 | | |

### c) Les opérateurs relationnels sur les entiers : restent valables.

| Syntaxe Algorithmique | = | ≠ | < | ≤ | > | ≥ |
|---|---|---|---|---|---|---|
| Syntaxe Python | == | != | < | <= | > | >= |

## 2) Le type réel (Float):

### a) Définition :

Une variable de type float est un nombre décimal avec virgule flottante (avec un point décimal)
Un float en python doit être écrit avec un point décimal. (Jamais avec une virgule)

### b) Les opérateurs applicables sur les réels :

| Opérateur | Syntaxe | | Exemple : |
|---|---|---|---|
| | Algorithme | Python | |
| Addition | + | + | 1.2+6.25 = ..... |
| Soustraction | - | - | 2.45-6.0 = -3.55 |
| Multiplication | * | * | 1.3 * 0.2 = ..... |
| Division | / | / | 5.5 / 2 = ..... |
| Puissance | Puissance | ** | 0.0**0.0 = 1.0 |

| | | | |
|---|---|---|---|
| Division entière | // | // | 10.0//2.0 = …….. |
| Reste de la division entière | % | % | 10.0%2.0 = …….. |

**c)** Les fonctions arithmétiques standards sur les entiers et les réels :

***NB :*** **Pour utiliser les fonctions ci-dessous, il faut importer les modules « math » et « random »**

| Nom de la fonction en algorithme | En Python | Rôle | Type du paramètre x | Type du résultat | Exemples | |
|---|---|---|---|---|---|---|
| **tronc(x)** | **Trunc (x)** | Retourne la valeur entière de x. | Réel | Entier | trunc (-2.575) = ……..<br>trunc (3.14)= …….. | |
| **Arrondi(x)** | **round (x)** | Arrondit une valeur réelle à l'entier le plus proche. | Réel | Entier | round (9.499) = ……<br>round(2.35) = …….. | |
| | **round (x,nb)** | Arrondi une valeur réelle à une valeur de nb chiffres après la virgule | Réel | Réel | round (73.123 , 2) = 73.12 | |
| **abs(x) ou fabs(x)** | **abs(x)** | Retourne la valeur absolue de x | Réel ou Entier | Type de x | abs(-5) = ……. | abs(-5.44) = …… |
| **puissance(x,y)** | **pow(x,y)** | Retourne la puissance de x par y | Réel \|\| entier | Réel | pow(2.2 , 3) = …… | pow(10 , -2) = ……. |
| **RacineCarré(x)** | **sqrt(x)** | Retourne la racine carré de x (x>=0) | Entier ou Réel | Réel | sqrt (4) = …… | |
| **Aléa ( )** | **random( )** | Retourne une valeur réelle de [0 , 1 [ | - | Réel | x= random( ) x = …… | |
| **Aléa (x,y)** | **randint(x,y)** | Retourne une valeur entière entre x et y (avec x<=y) | Entier | Entier | x= randint(2,5) x = ……<br>x= randint(2,10) x = …….. | |

**d)** Les opérateurs d'assignation sur les entiers et les réels :

| Opérateur | Expression | Signification |
|---|---|---|
| **+=** | **a+=3** | **a=a+3** |
| **-=** | **a-=3** | **a=a-3** |
| ***=** | **a*=2** | **a=a*2** |
| **/=** | **a/=2** | **a=a/2** |
| **%=** | **a%=2** | **a=a%2** |

***Evaluation :***

| Instruction | Evaluation | Type du résultat |
|---|---|---|
| a=2.25<br>a+=2.75 | a= …… | …… |
| a - =2 | a = …….. | …… |
| a = float("12.36")<br>b = a + 5 | a = ……<br>b= …… | …… |
| a= 15.23<br>x= str(a) | x=…….. | …… |
| a =27.0<br>b =a%2 +2.36 | b=…….. | …… |
| a = round(2.235)<br>b= pow(a, 2) | b=…….. | …… |

***Application1 :***

Réorganiser les lignes du code suivant pour que le programme affiche le montant à payer pour un élève qui a acheté **q** cahiers, sachant que le prix d'un cahier est **p=2.5** et qu'il a eu une remise de **10%** :

a) print ("le montant à payer est : " , m)

b) m= p*q

c) q=int (input("donner le nombre de cahiers : " ))

d) r= (10*m)/100

e) p = 2.500

f) m=m-r

| N° | Instruction python |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |

***Application 2 :***

Ecrire un algorithme puis un script Python qui permet de calculer et d'afficher la distance « d » entre deux points M et N dans un repère de coordonnées respectives (x, y) et (x1, y1).

Sachant que $d = \sqrt{(x1 - x)^2 + (y1 - y)^2}$

**3) Le type booléen (bool):**

**a)** Définition :

En Python, une variable de type **bool** peut prendre soit la valeur **True** (vrai) soit **False** (faux).

**b)** Les opérateurs logiques sur les booléens :

Python définit 3 opérateurs logiques :

| Syntaxe Algorithmique | Syntaxe en Python |
|---|---|
| NON | not |
| ET | and |
| OU | or |

Ci-dessous **la table de vérité** des opérations logiques de deux variables booléennes A et B :

| Variable | Type |
|---|---|
| A, B | Booléen |

| A | B | NON(B) | A ET B | A OU B |
|---|---|---|---|---|
| Faux | Faux | **vrai** | **Faux** | **Faux** |
| Faux | Vrai | **faux** | **faux** | **vrai** |
| Vrai | Faux | **vrai** | **Faux** | **vrai** |
| Vrai | Vrai | **faux** | **vrai** | **vrai** |

**c)** Ordre de priorité entre les opérateurs logiques :

| Opérateur | Opérations | Exemple | Priorité |
|---|---|---|---|
| **( )** | Parenthèses | __ | **1** |
| **Non** | Négation | Non (3>4) =vrai | **2** |
| **ET** | Conjonction | Non (3>2) ET (5 <7) = faux | **3** |
| **OU** | Disjonction | Non ( 4=4) OU (-1 < 5) = vrai | **4** |

*Remarque* :
- ✓ Les opérations mises entre parenthèses sont les plus prioritaires.
- ✓ Si deux opérateurs de même priorité se succèdent, la priorité s'accorde à celui qui est à gauche…vers la droite.

***Evaluation :***

A ← (2 < 5 OU 15 > 24) ET ( 0 < 1)
……
……
……

B ← 2 < 5 OU (15 > 24 ET 0 < 1)
……
……
……
……

C ← (Arrondi (0.05) = 0) **OU** ( (2**2) < 2 ) **ET** (7 DIV 3 > 2 )
……
……
……

*Exemples d'applications :*

Soit l'affectation en « Python » suivante : a, b, c = 6, 7, 42

Évaluer les instructions ci-dessous en donnant à chaque fois le résultat de Bx obtenu :

| Instructions | Evaluation |
|---|---|
| B1= (a == 6) | B1= ……. |
| B2= (a==42) | B2= ……. |
| B3= (a == 6 and b == 7) | B3= ……. |
| B4=( a == 7 and b == 7) | B4= ……. |
| B5= (not (a != 7) and (b <= 7) ) | B5= ……. |
| B6= (a == 7) or (c == 42) | B6= ……. |
| B7=(not (a == 7)) and (not (c != 6)) | B7= ……. |

**4) Le type chaîne de caractères : (str)**

## 1. Définition :

- Un ***caractère*** est un symbole unique, qui peut être une lettre minuscule "a", une lettre majuscule "B", un symbole spécial "&", un chiffre "7", un espace " ".
- Une *chaîne de caractères* (*string* en anglais) est une suite de caractères placée entre guillemets (simple ou double). Une chaîne de caractères peut contenir 0, 1 ou plusieurs caractères.

- ***Une chaine ne contenant aucun caractère est dite chaine vide. (ch='' ).***
- ***Une chaine représentée par un seul caractère est de longueur 1.***

***Exp : ch ='a' : ch est représentée par la lettre 'a'. ch = '1', ch ='#' , …***

- ***La chaine 'ali' est une chaine de longueur 3 (contient 3 lettres).***

**NB** : pour savoir la longueur d'une chaine, on peut utiliser la fonction len ( ) :

```
>>> ch = 'python'
>>> print ( len (ch) )
……
>>> ch1 = 'p'
>>> print ( len(ch1))
……
>>> ch2 = ''
>>> print ( len(ch2))
……
```

***Remarque :*** une **chaîne de caractères** est un objet immuable (c-à-d ***non modifiable***) ;
ni ***changement***, ni ***suppression*** et ni ***insertion***.

## 2. Opérations sur les chaînes de caractères

### a. *Concaténation & multiplication :*

***Activité : C****ompléter le script ci-dessous :*

```
>>>  x= 'Mohamed  '
>>> y = 'Ali'
>>> x+y                    # concaténer deux chaines (str)
……..
>>> y = 3
>>> x * y                  # multiplier (ou répéter) une chaine y fois
……..
```

### b. *Les Fonctions standards sur les chaines :*

**Activité N°1 :** lancer IDLE Python, saisir la séquence d'affectations suivante et compléter le tableau ci-dessous :

```
ch="langage"
c="a"
ch1="python"
 x = "2020"
y = 52
ph="13.75"
ch2= "4201"
```

| Fonction en Python | Résultat à afficher | Rôle |
|---|---|---|
| a = ord(c) | | Renvoie le code ASCII du caractère c |
| b = chr(a) | | Renvoie le caractère dont le code ASCII est a |
| y = str(y) | | Convertit y en une chaine de caractère |
| n = int(ch2) | | Convertit la chaîne ch2 en un nombre entier |
| r = float(ph) | | Convertit la chaîne ph en un nombre réel |
| l = len(ch) | | Renvoie la longueur de ch |
| mi = min(ch) | | Renvoie le caractère ayant le plus petit Code ASCII dans ch |
| ma = max(ch) | | Renvoie le caractère ayant le plus grand Code ASCII dans ch |
| w= choice(ch) | | Renvoie, au choix, un caractère de ch.<br>➡ il faut importer le module random |

## c. Les Méthodes de recherche sur les chaines :

| | | |
|---|---|---|
| nb=ch.count(c) | | Renvoie le nombre d'occurrences de c dans ch |
| ind=ch.index(c) | | Renvoie l'indice de c dans ch |
| p=ch.find(c) | | Renvoie la position de c dans ch |

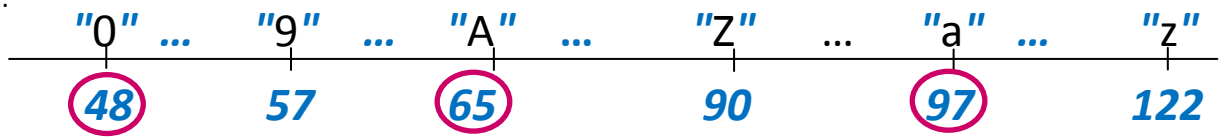
## d. Les méthodes de changement de casse sur les chaines :

| | | |
|---|---|---|
| ch.upper() | | Convertit ch en Majuscule |
| ch.lower() | | Convertit ch en Minuscule |
| ch.capitalize() | | Convertit en Majuscule la 1ère lettre de ch |
| ch=ch+" "+"scientifique"<br>ch.title() | | Convertit en Majuscule la 1ère lettre de chaque mot dans ch |
| ch1.replace('p', 'P') | | Remplace chaque occurrence de 'p' par 'P' dans ch1 |
| '*'.join(ch) | | Sépare les caractères de ch avec *<br>➥On peut utiliser d'autres séparateurs. Exp :'-', '/', … |

***Remarque*** :

**1)** Les deux fonctions **chr** et **ord** s'appliquent aux caractères et permettent respectivement de retourner le caractère dont le code ASCII est donné et le code ASCII d'un caractère donné.

Tel que :




**2)** La comparaison entre deux chaines de caractères se fait caractère par caractère selon leur code ASCII.

*Exp ;* print ("bonjour">"BONJOUR") # affiche ……

print ("formatage" >"formation") # affiche ……

**3)** print ( **chr** ( **ord**("!") ) ) # affiche "!"

print ( **ord** ( **chr**(130) ) ) # affiche 130

## e. Les Méthodes de vérification :

Activité N° 2 : Ci-dessous le début d'un script python incomplet :

```
ch="informat-2020"
c="a"
ch1="TI"
x = "1234"
```

➥ *Compléter le script en exécutant toutes ces opérations afin d'en déduire le résultat :*

| Méthode | Résultat à afficher | Rôle |
|---|---|---|
| ch.**isalnum**( ) | | Renvoie True si ch est alphanumérique sinon False |
| c.**isalpha**( ) | | Renvoie True si ch est alphabétique sinon False |

| | | |
|---|---|---|
| x.**isdigit**( ) | | Renvoie True si ch est numérique sinon False |
| ch1.**islower**( ) | | Renvoie True si ch est en Minuscule sinon False |
| ch1.**isupper**( ) | | Renvoie True si ch est en Majuscule sinon False |
| ch.**isspace**( ) | | Renvoie True si la chaine est formée uniquement par des blancs sinon False. |

## 3. L'accès à un caractère d'une chaîne :

- Une chaîne de caractères est une séquence ***indexée*** de caractères.
- Les caractères d'une chaine sont indexés à partir de 0.
- Pour accéder au ième caractère d'une chaîne, il suffit de donner le nom de la chaîne suivi de l'indice « i » entre 2 crochets et se note : **ch [ indice ]**

| ch = | B | O | n | j | o | u | r |
|---|---|---|---|---|---|---|---|
| **Indice positif** | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| **Ou Bien** | | | | | | | |
| **Indice négatif** | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

***Exemple*** : ch[0] : est le 1er caractère (le plus à gauche) de la chaine ch.

Activité N°3 : Soit ch une variable de type chaîne de caractère et C un caractère.

ch = "Bonjour"

c = ch[0] c = ‘’….. ‘’ x= ord(ch[0]) =…….

c = ch [3] ; c = ‘’ …..’’ y= chr(ord(ch[3]))=… ……

c = ch[-1] ; c = ‘’….. ‘’

c = ch[**len**(ch)-1] c = ‘’….. ‘’

## 4. Les sous-chaînes [slicing]:

Il est possible de découper une chaine en tranches. Une tranche est une ***sous-chaine*** qui commence par l'indice ***début*** et se termine par l'indice ***fin-1*** et se note : **ch [début : fin].**

***Activité 1*** : *Exécuter ces instructions en Shell Python, puis donner le résultat avec :*

*ch* = ''le loup lape le lait''

| ***Instructions*** | ***Résultat à afficher*** |
|---|---|
| ch[3 :7] | |
| ch[ 8 :12 ] | |
| ch[ 0 :-2] | |
| ch[3 :] | |
| ch[-1 :] | |
| ch[ -4 :] | |
| ch[ -7 :] | |
| ch[ : :-1] pas=-1 | |
| ch[ : :-2] pas=-2 | |
| ch [ :] | |

***Complément*** *: **Appartenance d'un caractère dans une chaine.***

*Exécuter ces instructions en Python, puis donner le résultat :*

```
>>> ch='python'
>>> c='p'
>>> print(c in ch)
.....
>>> print(c not in ch)
.....
```

**in**
**not in**

Vérifient l'appartenance (existence) d'une 1ère chaine dans une deuxième.