

## Les Entrées / Sorties

```
# commentaire sur une ligne : la valeur sera saisie sous forme de chaîne de caractères
x = input("saisir une valeur : ")
print("valeur = ",x)
```

.....

**début du commentaire** : ceci est un commentaire sur plusieurs lignes

Pour obtenir un entier, il faut convertir la saisie en un nombre

Exemple : val\_x\_entier = int(x)

**fin du commentaire**

.....

## Les types de variables

int : nombre entier (ex : 18 0b1010 0xF4)

float : nombre décimal (ex : 3.56 5.2e-3)

str : chaîne de caractères

## Les opérations

| Opérations                | Symboles | Exemples        |
|---------------------------|----------|-----------------|
| addition                  | +        | 2 + 5 donne 7   |
| soustraction              | -        | 8 - 2 donne 6   |
| multiplication            | *        | 6 * 7 donne 42  |
| puissance                 | **       | 2 ** 3 donne 8  |
| division                  | /        | 7 / 2 donne 3.5 |
| division entière          | //       | 7 // 2 donne 3  |
| reste de division entière | %        | 7 % 2 donne 1   |

| Symboles | Opérateurs        |
|----------|-------------------|
| >        | supérieur         |
| <        | inférieur         |
| >=       | supérieur ou égal |
| <=       | inférieur ou égal |
| ==       | égal              |
| !=       | différent         |

| Opérations logiques | Fonctions  |
|---------------------|------------|
| and                 | ET logique |
| or                  | OU logique |

## Les tableaux

La notion de tableau n'existe pas en Python. On utilise les **listes** qui peuvent contenir à la fois des entiers, des flottants, des chaînes de caractères, des listes.

```
liste1 = [1.56, "tabouret", 3]
```

On accède à un élément par son indice (1<sup>er</sup> indice=0) :

```
>>> liste1[1]
'tabouret'
```

De nombreuses opérations existent avec les listes. En voici quelques unes :

*append* : ajouter un élément en fin de liste

*insert* : insérer un élément dans une liste

*remove* : supprimer un élément

*+* : concaténer 2 listes

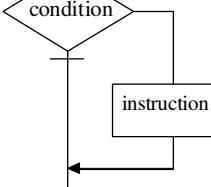
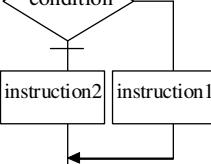
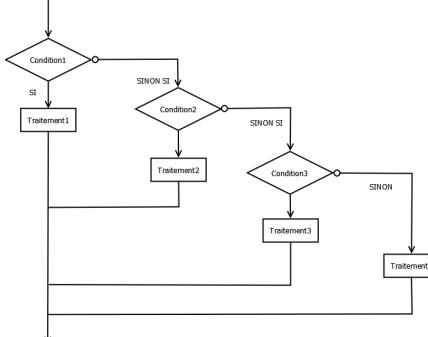
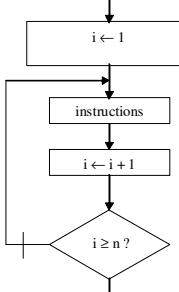
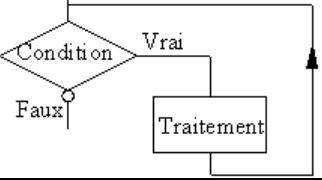
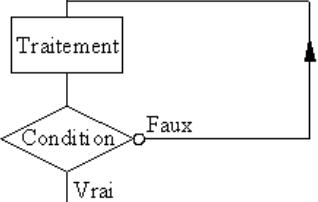
*len* : nombre d'éléments de la liste >>> N1 = len(liste1)

```
>>> print(N1)
```

3

On peut également utiliser des listes à 2 dimensions : a[ligne][colonne]

## Les instructions

| algorithme  | algorigramme   | python   |
|---|--|--|
| si condition alors<br>instruction<br>fin si   |  <pre> graph TD     Cond{condition} --&gt; Instruction[instruction]     Instruction --&gt; Cond   </pre>  | if x>0 : print(x," positif")<br><br>ou bien<br><br>if x>0 :<br>print(x," positif")               |
| si condition alors<br>instruction1<br>sinon<br>instruction2<br>fin si   |  <pre> graph TD     Cond{condition} --&gt; Instruction1[instruction1]     Cond --&gt; Instruction2[instruction2]     Instruction1 --&gt; Merge(( ))     Instruction2 --&gt; Merge     Merge --&gt; Cond   </pre>  | if x>0 :<br>print(x," positif")<br>else :<br>print("x négatif ou nul")                           |
| si condition1 alors<br>instruction1<br>sinon si condition2 alors<br>instruction2<br>sinon<br>instruction3<br>fin si |  <pre> graph TD     Cond1{Condition1} -- SI --&gt; Traitement1[Traitement1]     Cond1 -- SINON SI --&gt; Cond2{Condition2}     Cond2 -- SI --&gt; Traitement2[Traitement2]     Cond2 -- SINON SI --&gt; Cond3{Condition3}     Cond3 -- SI --&gt; Traitement3[Traitement3]     Cond3 -- SINON --&gt; Traitement4[Traitement4]     Traitement1 --&gt; Merge1(( ))     Traitement2 --&gt; Merge1     Traitement3 --&gt; Merge1     Traitement4 --&gt; Merge1     Merge1 --&gt; Cond1   </pre> | if x>0 :<br>print(x," positif")<br>elif x<0 :<br>print(x," négatif")<br>else :<br>print("x nul") |
| pour i<-1 jusqu'à n faire<br>instructions<br>fin pour   |  <pre> graph TD     Init[i ← 1] --&gt; Instructions[instructions]     Instructions --&gt; Update[i ← i + 1]     Update --&gt; Cond{i ≥ n ?}     Cond -- Faux --&gt; Init     Cond -- Vrai --&gt; Traitement[Traitement]     Traitement --&gt; Cond   </pre>   | for i in range(0,5) :<br>print("i=",i)   |
| tant que condition faire<br>instructions<br>fin tant que  |  <pre> graph TD     Traitement[Traitement] --&gt; Cond{Condition}     Cond -- Vrai --&gt; Traitement     Cond -- Faux --&gt; Exit   </pre>  | i=10<br>while i!=0 :<br>print("i=",i)<br>i=i-2   |
| faire<br>instructions<br>tant que condition   |  <pre> graph TD     Cond{Condition} -- Vrai --&gt; Traitement[Traitement]     Traitement --&gt; Cond     Cond -- Faux --&gt; Exit   </pre>  | Pas de do...while en Python  |