Instruction conditionnelle

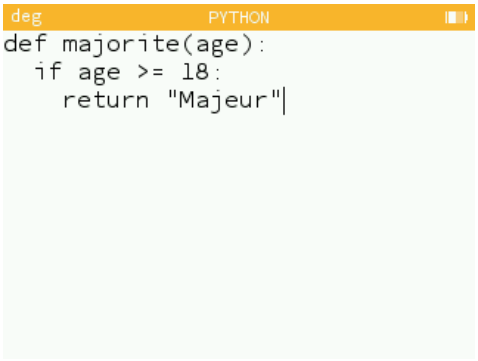
Au sein d'une fonction, il peut être intéressant d'exécuter des instructions sous certaines conditions.

If … elif … else

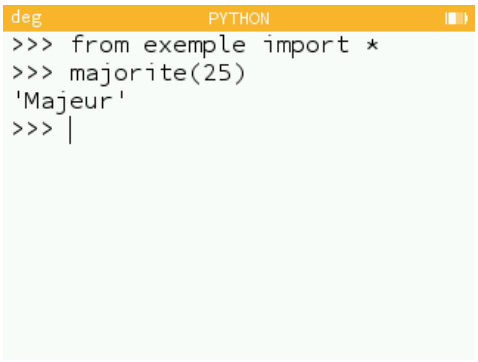
Par exemple, si vous souhaitez écrire une fonction `majorite(age)` qui renvoie `"Majeur"/"Mineur"` selon l'âge renseigné, il est nécessaire de séparer les cas où l'âge est supérieur à 18 et celui où il est inférieur.
En Python, un bloc d'instruction conditionnelle s'écrit en indiquant :

```
if condition:
    instruction
```

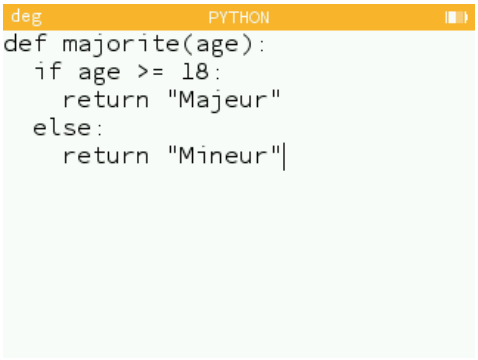
L'instruction n'est exécutée que si la condition suivant `if` est vérifiée. Les deux points servent à indiquer à Python que vous commencez un bloc d'instructions.
Pour notre exemple, on peut écrire :



La fonction ci-dessus répond donc le texte "Majeur" si l'âge indiqué entre parenthèses est supérieur ou égal à 18.
Voir l'application ci-dessous :



Si l'on souhaite ajouter la possibilité de répondre `"Mineur"`, il faut ajouter un cas. On souhaite que la fonction réponde `"Majeur"` si l'âge est supérieur à 18 et `"Mineur"` sinon. Pour signifier ce SINON à Python, on utilise `else` suivi de deux points avec la même indentation que le premier `if`.



En résumé :

— S'il n'y a qu'un seul cas à distinguer, on utilisera :

```
if condition:
    instruction
```

— S'il n'y a que deux cas à distinguer, on utilisera :

```
if condition:
    instruction_1
else:
    instruction_2
```

— S'il y a plus de deux cas, on utilisera `elif` pour ajouter des conditions :

```
if condition_1:
    instruction_1
elif condition_2:
    instruction_2
elif condition_3:
    instruction_3
```

Les conditions

Voici la syntaxe des conditions dans Python :

Condition	Syntaxe Python
Si x est égal à y	`if x==y:`
Si x est différent de y	`if x!=y:`
Si x est strictement supérieur à y	`if x>y:`
Si x est strictement inférieur à y	`if x<y:`
Si x est supérieur ou égal à y	`if x>=y:`
Si x est inférieur ou égal à y	`if x<=y:`

Si l'on souhaite vérifier deux conditions, on utilise **and** entre les deux conditions. Par exemple :

```
if x==1 and y>0:
```

Si l'on souhaite vérifier l'une ou l'autre des deux conditions, on utilise **or** entre les deux conditions. Par exemple :

```
if x>=1 or x==0:
```

Appuyez sur la touche Paste de votre calculatrice pour faire apparaître un menu de raccourcis. Dans **Boucles et tests** vous trouverez des blocs d'instructions pré-remplis pour vous éviter d'écrire lettre par lettre au clavier.

Exercice

Écrire une fonction **vabsolue(x)** qui prend un réel en argument et renvoie sa valeur absolue.

Analyse de l'énoncé

La valeur absolue d'un nombre x, notée $|\ x\ |$, est sa distance à zéro. La valeur absolue d'un réel positif est donc ce nombre et celle d'un nombre négatif est son opposé.

En formalisant on peut écrire :

- $|\ x\ | = x$ si x est positif ;
- $|\ x\ | = -x$ si x est négatif.

On souhaite écrire une fonction qui prend un réel x en argument et qui renvoit sa valeur absolue. Il va donc falloir effectuer un test sur le signe de x pour donner le résultat.

Résolution

On voit ici apparaître une instruction conditionnelle qui distingue deux cas :

- Si x est positif, renvoyer x ;
- Sinon, renvoyer $-x$.

Le cas $x = 0$ peut être inclus dans l'un ou l'autre des cas. On peut donc écrire au choix : `if x>0:` ou bien `if x>=0:`.

```
deg PYTHON
def vabsolue(x):
  if x > 0:
    return x
  else:
    return -x
```

```
deg PYTHON
>>> from exercice import *
>>> vabsolue(5)
5
>>> vabsolue(-5)
5
>>> vabsolue(0)
0
>>>
```

Remarque pour finir

La fonction valeur absolue est déjà présente dans Python. Il suffit d'écrire `abs()` pour l'appeler.

```
>>> abs(-4)
4
```

Autre exercice

Écrire une fonction **`mediane`** qui prend une série de nombres de taille quelconque en argument et qui renvoie la médiane de la série.

```
deg PYTHON
def mediane(liste):
  liste=sorted(liste)
  n=len(liste)
  if n%2==1:
    return liste[int((n-1)/2)]
  else:
    v_1=liste[int(n/2-1)]
    v_2=liste[int((n/2))]
    return (v_1+v_2)/2
```

Quelques remarques sur le corrigé

La commande **`sorted(liste)`** renvoie la liste des valeurs classées par ordre croissant :

```
>>> sorted([5,3,4])
[3,4,5]
```

Le caractère **`%`** (modulo) donne le reste de la division euclidienne. Ainsi **`effectif%2`** teste si **`effectif`** est pair ou impair.

```
>>> 5%2
1
```

L'indice d'un élément d'une liste doit être entier. C'est pour cela que l'on utilise **`int`** (*integer* = "entier" en anglais) entre les crochets :

```
>>> int(4.0)
4
```

Pour sélectionner le k-ième élément d'une liste L, on écrit **`L[k-1]`**. En effet, le premier indice d'une liste est 0 en Python. On sélectionne ici l'élément placé en (n-1)/2 dans **`liste`**, le 3ème élément pour une liste à 5 éléments :

```
>>> liste = [1,4,7,9,10]
>>> n = 5
>>> liste[int((n-1)/2)]
7
```