



# Les tests

- => Les tests permettent à l'ordinateur de prendre des décisions si telle ou telle condition est vraie ou fausse.
- => Python utilise l'instruction **if** ainsi qu'une comparaison que nous avons abordée au chapitre précédent

```
x = 2
if x == 2:
... print("Le test est vrai !")
```

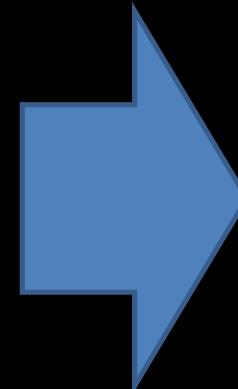
...

Le test est vrai !



## Syntaxe générale

```
if condition1 :  
    instruction1  
    instruction2  
    ...  
else :  
    instruction3  
    instruction4  
    ...
```



```
>>> x = 2  
>>> if x == 2:  
...     print("Le test est vrai !")  
... else:  
...     print("Le test est faux !")
```

**important**

Attention à l'indentation et les deux points



## Tests à plusieurs cas

*tester si la condition est vraie ou si elle est fausse dans une même instruction if*

*On utilise souvent des opérateurs logiques*



## L'opérateur "OU" "or"

important

Respectez bien la  
casse des opérateurs  
and et or qui, en  
Python, s'écrivent en  
minuscule

Condition 1	Opérateur	Condition 2	Résultat
Vrai	OU	Vrai	Vrai
Vrai	OU	Faux	Vrai
Faux	OU	Vrai	Vrai
Faux	OU	Faux	Faux

## L'opérateur "ET" "and"

Condition 1	Opérateur	Condition 2	Résultat
Vrai	ET	Vrai	Vrai
Vrai	ET	Faux	Faux
Faux	ET	Vrai	Faux
Faux	ET	Faux	Faux



## Opérateurs logiques

```
>>> x = 2
>>> y = 2
>>> if x == 2 and y == 2:
... print("le test est vrai")
...
le test est vrai
```

## Imbrication de tests

```
>>> x = 2
>>> y = 2
>>> if x == 2:
... if y == 2:
... print("le test est vrai")
...
le test est vrai
```



## Tests de valeur sur des réels



```
>>> (3 - 2.7) == 0.3
```

False

```
>>> 3 - 2.7
```

0.2999999999999998

Nous voyons que le résultat de l'opération  $3 - 2.7$  n'est pas exactement  $0.3$  !!!!!! d'où le **False**. Pour éviter ces problèmes nous conseillons de toujours d'arrondir vos résultats





## Arrondir un nombre réel

```
>>> round(3.1415)  
3
```

## Arrondir un nombre réel au dixième

```
>>> x = 1.4567  
>>> round(x,1)  
1.5
```

Argument à modifier (nb de chiffres après la virgule)



## Convertir un nombre réel en entier

```
>>> x = 3.1415
>>> x = int(x)
>>> x
3
>>> type(x)
<class 'int'>
```



## Arrondir à l'entier sup ou inf

```
>>> x = 3.1415  
>>> x = ceil(x)  
>>> x  
4
```

```
>>> x = 3.1415  
>>> x = floor(x)  
>>> x  
3
```

```
>>> x = -3.1415  
>>> x = floor(x)  
>>> x  
-4
```



## Questionnaire

- 1- Donnez la syntaxe générale de l'instruction conditionnelle en Python
- 2- Donnez la signification des opérateurs "and" et "or"
- 3- Quelle est le rôle de l'indentation
- 4- Quelle instruction permet
  - a. d'arrondir un chiffre avec virgule?
  - b. De convertir un nombre réel en entier?



## Travaux pratiques

Proposez un script qui permet de vérifier si le nombre rentré par un utilisateur est pair ou impair ou nul avec deux blocs "if" imbriqués

[solution page suivante](#)



```
x=input("rentrez un nombre")  
  
x=int(x)  
  
if x !=0 :  
  
    if x%2==0:  
  
        print ("pair")  
  
    else:  
  
        print("impair")  
  
    else:  
  
        print("nul")
```