Séance 3

LES TESTS

L'essentiel

- Les **booléens** (`True` ou `False`) vont permettre de traduire des conditions.
- Tests de comparaison :

Syntaxe Python	`x<y`	`x>y`	`x==y`	`x!=y`	`x<=y`	`x>=y`
Signification	$x < y$	$x > y$	$x = y$	$x \neq y$	$x \leqslant y$	$x \geqslant y$

- **Attention** : il faut bien distinguer le symbole d'affectation = du symbole de comparaison ==.
- On peut ensuite combiner les booléens entre eux avec des opérateurs logiques : `and` (et), `or` (ou) et `not` (négation).

```
>>> 1 > 2
False
```

```
>>> 1+3 == 4
True
```

```
>>> 3<=8 and 3>0
True
```

```
>>> 10<0 or 10==5
False
```

```
>>> not(8**2 == 64)
False
```

- **SI...ALORS** : Pour n'exécuter des instructions que sous certaines conditions, on utilise en Python la structure suivante :

```
if conditions :
    instructions
```

- **SI...ALORS...SINON** : Lorsqu'on veut gérer une disjonction de cas à 2 issues, on utilise la structure suivante :

```
if conditions :
    instructions1
else :
    instructions2
```

- **DISJONCTIONS DE CAS** : Dans les cas où il y a plus de deux choix possibles, on peut utiliser une version plus élaborée :

```
if conditions1 :
    instructions1
elif conditions2 :
    instructions2
  ...
elif conditionsa :
    instructionsa
else :
    instructionsf
```

- Attention, penser :
  - ◇ Aux **deux points** en fin de ligne.
  - ◇ À **indenter** comme pour les fonctions l'ensemble des instructions à effectuer dans le SI.
  - ◇ Après un `else`, il n'y a jamais de conditions. Il correspond à tous les cas ne vérifiant pas les conditions précédentes.
  - ◇ Il n'est pas obligatoire d'avoir l'instruction `else` dans une structure conditionnelle.

Exercices du jour

Exercice 1 : Corriger chacune des fonctions suivantes, afin qu'elles réalisent ce qui est demandé :

1. Renvoie `True` si le nombre entré par l'utilisateur est pair `False` sinon.

```
def pair(n):
if n%2=0
return True
```

2. Reçoit deux nombres réels et vérifie s'ils sont de même signe :

```
def signes(x,y) :
if xy>0 :
    return "Même signe"
    else
    return "Signes différents"
```

Exercice 2 : On considère la fonction ci-dessous :

```
def jeu(x) :
    if x**2 < 4 and x <= -1 :
        return "Gagné"
    else :
        return "Perdu"
```

1. Trouver une valeur de `x` telle que `jeu(x)` renvoie "Gagné".
2. Trouver (papier/crayon) l'ensemble des réels `x` tels que `jeu(x)` renvoie "Gagné"..
3. ★ Écrire une fonction qui se comporte exactement comme `jeu`, mais qui n'utilise pas de `else`.

Exercice 3 : On considère l'équation $ax^2+bx+c=0$. Écrire une fonction `equation` qui reçoit les coefficients `a`, `b` et `c`, et qui renvoie le nombre de solutions réelles de cette équation.

Exercices en autonomie

Exercice 4 : Créer une fonction `entier(x)` qui reçoit un flottant `x` et renvoie un booléen indiquant si ce nombre est entier ou non.

Exercice 5 : Créer une fonction qui reçoit un nombre entier naturel. Cette fonction ne renvoie rien, mais affiche si ce nombre est divisible par 2, divisible par 3, divisible par 5, divisible par 10. *Bien sûr un nombre peut être divisible par plusieurs entiers...*

Aide pour les exercices

Indication 4 conseil : penser à utiliser l'instruction `round`

Solutions des exercices

Correction 4 Une solution :

```
def entier(x) :
    if round(x) == x :
        return True
    else :
        return False
```

Ou plus court :

```
def entier(x) :
    return round(x) == x
```

Correction 5 On test un à un :

```
def diviseurs(n) :
    if n % 2 == 0 :
        print(n, "est divisible par 2")
    else :
        print(n, "n'est pas divisible par 2")
    if n % 3 == 0 :
        print(n, "est divisible par 3")
    else :
        print(n, "n'est pas divisible par 3")
    if n % 5 == 0 :
        print(n, "est divisible par 5")
    else :
        print(n, "n'est pas divisible par 5")
    if n % 10 == 0 :
        print(n, "est divisible par 10")
    else :
        print(n, "n'est pas divisible par 10")
```

Bien sûr, on a envie d'éviter tous ces copier / coller, alors on peut avoir recours à une fonction auxiliaire :

```
def teste1diviseur(n, d):
    if n % d == 0 :
        print(n, "est divisible par", d)
    else :
        print(n, "n'est pas divisible par", d)

def diviseurs(n) :
    teste1diviseur(n, 2)
    teste1diviseur(n, 3)
    teste1diviseur(n, 5)
    teste1diviseur(n, 10)
```