Structures conditionnelles

Afin d'exécuter un code seulement si une condition est vérifiée, il existe deux possibilité syntaxique : un bloc conditionnel et l'opérateur ternaire.

Conseil

> La maîtrise de l'opérateur ternaire n'est pas exigée dans ce cours. Il est possible que nous la rencontrions car elle peut être élégante mais il est toujours possible de la remplacer par un bloc (qui fera 4 lignes au lieu d'une seule...).

Structure si/sinon

Python fournit une structure de **si** qui permet de réaliser des disjonctions : si le booléen (en général un test logique) est vrai, on effectue telles instructions, sinon on exécute tel autre bloc d'instructions.

In [1]:
```
n = int(input("Saisissez un entier : "))
if n % 2 == 0:  # On teste si n modulo 2 est nul
    # Instructions à faire si le booléen est vrai (donc quand n est pair)
    print(f"{n} est pair !")
print("C'est fini.")
```

```
Saisissez un entier : 36
36 est pair !
C'est fini.
```

- le code sait quand finit le bloc de code à réaliser si la condition est vrai grâce à l'**indentation**. Il n'y a ni mot clef *end* ni accolade.
- il ne faut pas oublier les 2 points après le booléen (ou la condition).

En python, l'indentation est **synthaxique**. Nous venons de voir avec le `if` que c'est elle qui sert à délimiter le bloc de code à exécuter si la condition est vraie mais cela est plus général : l'indentation sert toujours à délimiter un bloc de code. Elle n'est donc pas facultative ; si vous indentez mal votre code, il sera **faux**. Au passage, remarquer que le symbole `#` permet de faire des commentaires.

Il est aussi possible de préciser quoi faire si la condition est fausse :

In [2]:
```
n = int(input("Saisissez un entier : "))
if n % 2 == 0:  # On teste si n modulo 2 est nul
    # Instructions à faire si le booléen est vrai (donc quand n est pair)
    print(f"{n} est pair !")
else:
    # Instructions à faire si le booléen est vrai (donc quand n est pair)
    print(f"{n} est impair !")
```

```
Saisissez un entier : 45
45 est impair !
```

Voire d'emboiter les conditionnelles :

In [8]:
```
n = int(input("Saisissez un entier : "))
if n % 3 == 0:
    print(f"{n} est divisible par 3.")
elif n % 3 == 1:
    print(f"{n} est de la forme 3n+1.")
else:
    print(f"{n} est de la forme 3n+2.")
```

```
Saisissez un entier : 24525
24525 est divisible par 3.
```

Remarque :

- Attention, `a == 0` est un test d'égalité mais a=0 est une affectation ! De plus, `a = 1` peut renvoyer vrai dès que l'affectation a bien été réalisée.
- Il faut éviter les test `if A == True:` (écrire directement `if A:` , cela fera la même chose).

Opérateur ternaire

Il est parfois pratique lorsque les instructions conditionnelles à faire sont courtes d'utiliser l'opérateur ternaire. La syntaxe est la suivante

```
instruction_Vrai if condition else instruction_Fausse
```

Voici un exemple

In [1]:
```
x = int(input("Entrez un nombre entier : "))
# opérateur ternaire
fx = 3*x+1 if x % 2 else x // 2
# qui est utilisé à la place de ce bloc
# if x%2 == 1:
#     fx = 3*x+1
# else:
#     fx = x//2
print(f"f({x}) = {fx}")
```

```
Entrez un nombre entier : 1533567
f(1533567) = 4600702
```

In [4]:
```
x = 13
print(x % 2)
print(x % 2 == 1)
```

```
1
True
```

In [10]:
```
x = 139574728482
print(x % 17)
if not x % 17:
    print(f"{x} est congru à 0 modulo 17")
else:
    print(f"{x} n'est pas congru à 0 modulo 17")
```

```
0
139574728482 est congru à 0 modulo 17
```

In [ ]:
```

```