TP2

INSTRUCTION CONDITIONNELLE IF

À préparer : cours, exercices 1 et 2

Notions utilisées dans ce TP

Ce TP utilise les parties suivantes du vade-mecum (en plus du TP précédent).
— II : les types de variables → booléens
— III : bibliothèque random
— V : instruction if
— VI : fonctions

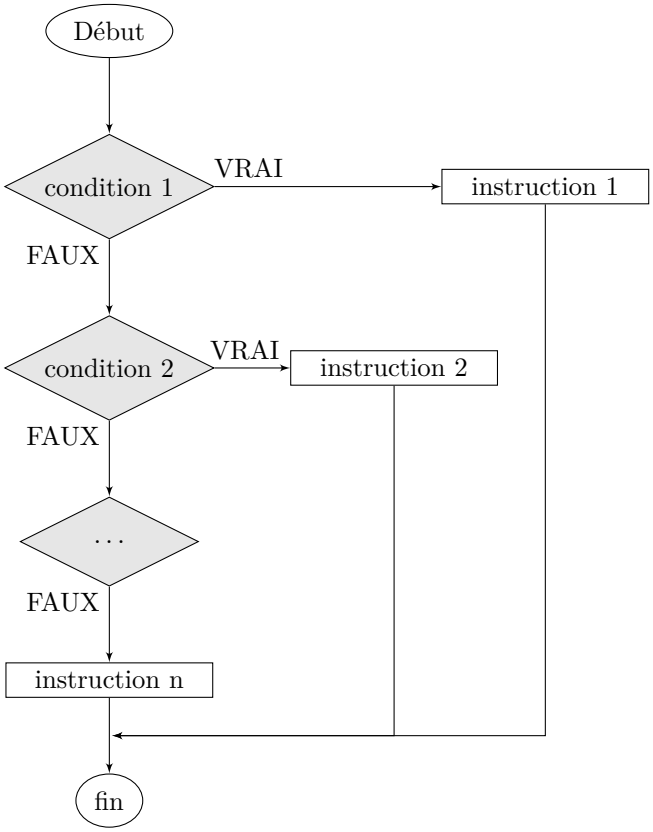
Nous allons voir comment écrire un programme python comportant une instruction conditionnelle du type **si ... alors ...**

Algorithme

Si condition 1,
    Faire l'instruction 1
Sinon, si condition 2,
    Faire l'instruction 2
...
Sinon
    Faire l'instruction n
Fin du Si

Programme python

```
if condition 1 :
    instruction 1
elif condition 2 :
    instruction 2
elif condition 3 :
    instruction 3
...
else :
    instruction n
```



— Les *conditions* sont des booléens.
— **Ne pas oublier** :
  — les deux points qui indiquent la fin de chaque condition ;
  — l'indentation des blocs d'instructions par rapport à la ligne if, elif ou else dont ils dépendent ;
  — la fin de l'indentation indique la fin de l'instruction conditionnelle.

Exercice 1

1. Écrire une suite d'instructions en langage Python qui demande à l'utilisateur de saisir deux réels a et b, puis affiche lequel est le plus petit des deux ou s'il y a égalité.

```
```

2. Tester avec plusieurs valeurs *pertinentes* de a et b que l'on détaille ici :

Exercice 2 Second degré

Écrire une fonction prenant en entrée trois flottants a, b, c (avec $a \neq 0$) et renvoyant les éventuelles solutions réelles de l'équation $ax^2 + bx + c = 0$.
Indications :

- la fonction sqrt de la bibliothèque math renvoie la racine carrée
- on peut renvoyer plusieurs valeurs en écrivant `return valeur1, valeur2,...`
  Le résultat renvoyé est alors un tuple.
- dans le cas où il n'y a rien à renvoyer, on écrit `return None`
- utiliser des noms de variables *explicites* et ajouter des commentaires dès que nécessaire.

```
from math import *

def second_degre(a,b,c):
    '''second_degre(a:float, b:float, c:float) -> racines reelles de
    ax**2+bx+c=0'''
```

Proposer des équations permettant de tester tous les cas de figure.

Exercice 3 Écrire un programme qui demande à l'utilisateur de saisir un entier n puis, lorsque n est pair, calcule :

$$k = \begin{cases} \dfrac{n}{2} - 7 \text{ si } n \geqslant 14 \\ \dfrac{n}{2} + 7 \text{ sinon} \end{cases}$$

et, lorsque n est impair, calcule : $k = \dfrac{n+3}{2}$, puis enfin affiche k.

```

```

Proposer des valeurs de n permettant de tester tous les cas de figure.

Exercice 4

1. Charger la fonction `randint` de la bibliothèque `random`, dont on donne ici la description :

```
randint(a:int, b:int) -> Renvoie un entier aléatoire N tel que a <= N <= b.
```

```
from random import randint
```

2. Comment simule-t-on le résultat d'un dé à 6 faces avec la fonction `randint` ? à 16 faces ?

```

```

3. Déterminer avec Python le gain final (mise de départ comprise, donc positif ou négatif) lors de ce jeu :
   Une personne joue avec un dé à 16 faces. Il mise en début de jeu une somme S (la mise est une somme donnée donc perdue). Si le résultat est pair et compris entre 1 et 5, il récupère 3 fois sa mise, si le résultat est pair et compris entre 6 et 10, il récupère 2 fois sa mise, si le résultat est pair et compris entre 11 et 16 alors il récupère sa mise et si le résultat est impair, il ne récupère rien.

```

```

4. Tester le programme, en s'assurant que tous les cas de figure ont été testés.

Exercice 5 Écrire une fonction qui, étant donnée une année, renvoie True si celle-ci est bissextile et renvoie False sinon.
Une année est bissextile (366 jours) si l'un des deux cas suivants se produit :
— l'année est divisible par 4 et non divisible par 100 ;
— l'année est divisible par 400.
Sinon, elle est non-bissextile (365 jours).

```
def bissextile(annee):
    '''bissextile(annee:int) -> bool, True si annee est bissextile,
    False sinon'''
```

Proposer un jeu de tests complet pour cette fonction, permettant de vérifier tous les cas de figure.