1NSI	PROGRAMMATION	R
	Contrôle de flux d'exécution en Python	

Un programme Python est une suite ordonnée d'instructions. L'ordre dans lequel elles sont interprétées est appelé **flux d'exécution**.

Le **flux d'exécution** peut être modifié par des instructions particulières appelées i**nstructions de contrôle de flux d'exécution**. Il en existe deux types :

- instructions **alternatives**,
- instructions **itératives**.

Ces instructions délimitent alors des **blocs** d'instructions.

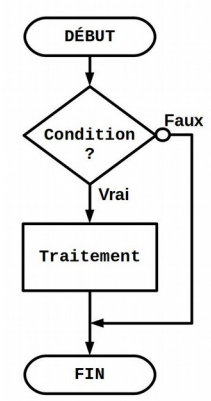
A retenir

Instructions et **blocs d'instructions** Python sont définis par la mise en page, en particulier par **l'indentation** (décalage) d'un multiple de 4 espaces afin de forcer le programmeur à prendre de bonnes habitudes.

I - LES STRUCTURES ALTERNATIVES PYTHON

I.A - STRUCTURE ALTERNATIVE RÉDUITE SI..ALORS

Algorigramme



Algorithme

```
DÉBUT
  SI <Condition> ALORS
    <Traitement>
FIN
```

Code Python exemple

```
i = 4
if i == 0 :
    print("i est nul")
```

- **Saisir** et **tester** le code Python proposé.
- **Modifier** le code pour que le programme affiche "i est nul".
- **Modifier** la condition du test, en utilisant l'opérateur **not** à la place de l'opérateur **==**, de sorte que le comportement du programme soit le même.

Rappel :

- toute variable ***nulle*** ou ***vide*** est considérée comme ***fausse***,
- toute variable ***non nulle*** ou ***non vide*** est considérée comme ***vraie***.

L'instruction `i = 4` a pour effet de stocker dans une cellule de la mémoire, étiquetée `i`, la valeur `4`. On souhaite maintenant que le programme propose l'utilisation de l'*interface d'entrée* de Python pour que l'utilisateur puisse interférer plus facilement avec le programme :

- **Remplacer** l'instruction `i = 4` par :

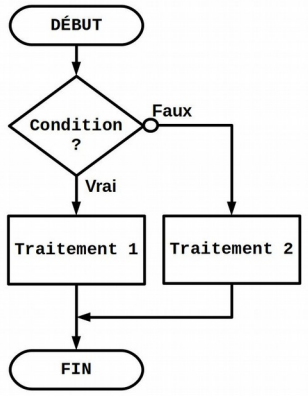
```
i = int(input("Saisir un entier : "))
```

Remarque : `"Saisir un entier : "` est une *invite*. C'est un message destiné à l'utilisateur pour l'aider dans sa décision.

- **Tester** le programme pour différentes valeurs de **i** saisies au clavier.

I.B - STRUCTURE ALTERNATIVE ÉTENDUE SI..ALORS..SINON

Algorigramme



Algorithme

```
DÉBUT
  SI <Condition> ALORS
    <Traitement 1>
  SINON
    <Traitement 2>
FIN
```

Code Python exemple

```
i = 4
if i < 0 :
    print("i est négatif")
else :
    print("i est positif")
```

- **Saisir** et **tester** le code Python proposé.
- **Modifier** la 1ère instruction pour que le programme affiche "i est négatif".
- **Modifier** le code pour qu'il affiche "i est négatif ou nul".
  - Modifier la condition du test,
  - Modifier le message concerné.

On souhaite que le programme utilise l'interface d'entrée de Python :

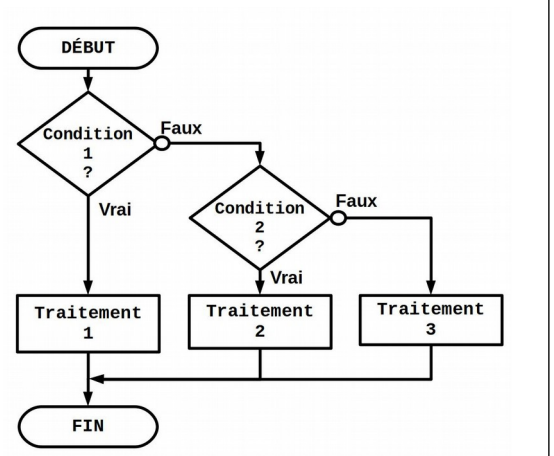
- **Remplacer** l'instruction `i = 4` par :

```
i = int(input("Saisir un entier : "))
```

- **Tester** le programme pour différentes valeurs de **i** saisies au clavier.

I.C - STRUCTURES ALTERNATIVES IMBRIQUÉES

Algorigramme	Algorithme

	**DÉBUT**
	  SI <Condition 1> **ALORS**
	  <Traitement 1>
	  SINON
	  SI <Condition 2> **ALORS**
	  <Traitement 2>
	  SINON
	  <Traitement 3>
	FIN

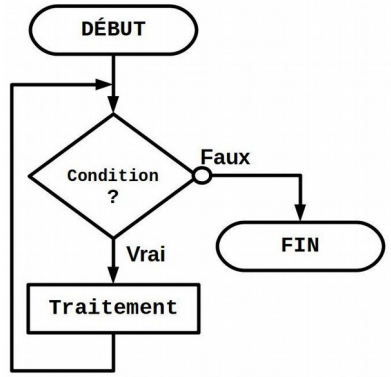
Code Python exemple

```
i = int(input("Saisir un entier : "))
if i < 0 :
    print("i est négatif")
elif i == 0 :
    print("i est nul")
else :
    print("i est positif")
```

- **Saisir** et **tester** le code Python proposé pour différentes valeurs de **i** saisies au clavier.

II - LES STRUCTURES ITÉRATIVES PYTHON

II.A - STRUCTURE ITÉRATIVE TANT QUE..FAIRE

Algorigramme	Algorithme
	**DÉBUT** **TANT QUE** <Condition> **FAIRE** <Traitement> **FIN**

Code Python exemple

```
i = 4
while i > 0 :
    print(i)
    i = i - 1
```

- **Saisir** et **tester** le code Python proposé.
- **Modifier** la condition du test pour que le programme affiche aussi zéro.
- **Remplacer** l'instruction `i = 4` par :

```
i = int(input("Saisir un entier : "))
```

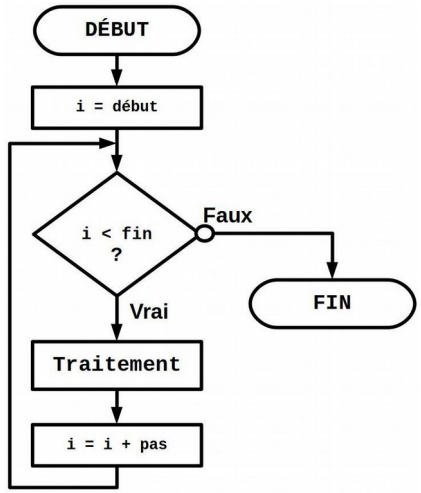
- **Tester** le programme pour différentes valeurs de **i** saisies au clavier.

Critère de choix de cette structure

Le nombre de répétitions n'est pas connu ou est variable.

II.B - STRUCTURE ITÉRATIVE POUR..FAIRE

Algorigramme



Algorithme

```
DÉBUT
  POUR <i> DE <début> À <fin>
    PAR PAS DE <pas> FAIRE
      <Traitement>
FIN
```

Code Python exemple

```
for i in range(0, 4, 1) :
    print(i)
```

- **Saisir** et **tester** le code Python proposé.

En Python, tout est objet, fonctions comprises. La fonction interne *range* a donc sa propre *classe.* Ses caractéristiques et ses comportements peuvent être consultés depuis un terminal Python en consultant l'*aide introspective de Python* : commande **`help(range)`**.

Après avoir consulté l'aide Python sur la fonction interne *range :*

- **Réduire** les trois paramètres de *range* à un seul en faisant en sorte que le programme se comporte de la même manière.
- **Modifier** les trois paramètres possibles de *range* pour que le programme affiche les entiers de 1 à 7 par pas de 2.

Critère de choix de cette structure

Le nombre de répétitions est connu et fixe.

III - EXERCICES

1. **Écrire** un programme permettant de vérifier si i appartient aux intervalles]-10,-1[ou]1,10], ou s'il est nul. Le programme renverra un de ces trois résultats :
   - i ∈]-10, -1[∪]1, 10]
   - i=0
   - i ∉]-10, -1[∪]1, 10] et i≠0
2. **Saisir** et **tester** le code suivant. **Modifier** le afin que le programme affiche 'secondes' au singulier lorsqu'il ne reste plus qu'une seconde à compter.

Compte à rebours de 5s

Algorithme

```
DÉBUT
  n = 5
  TANT QUE (n ≠ 0) FAIRE
    ÉCRIRE n secondes
    n ← n - 1
    ATTENDRE 1s
  ÉCRIRE Stop
FIN
```

Code Python exemple

```
import time

n = 5
while n != 0 :
    print(n, " secondes")
    n = n - 1
    time.sleep(1)
print("Stop.")
```

   - Aide : initialiser une variable *s*="secondes" qui deviendra *s*="seconde" lorsque n=1.

1. **Écrire** un programme qui affiche la valeur d'une variable qui s'incrémente de 0 à n inclus à l'aide d'une boucle **while**, n sera saisi en début de programme.
2. **Écrire** un programme similaire mais avec une boucle **for**.

Selon les critères de choix explicités précédemment, quelle est la structure de boucle à privilégier ? **Entourer** la bonne réponse.

for	while

3. **Écrire** un programme qui affiche la valeur d'une variable qui se décrémente de n inclus à 0 à l'aide d'une boucle **for**, n sera saisi en début de programme.
4. **Écrire** le programme 'compte à rebours 5s' précédent avec une boucle **for** et qui propose à la fin de recommencer ou d'arrêter. Dans le second cas le programme affiche "Au revoir".
   - Gestion du singulier de "secondes" demandé.