FICHE N°4 : INSTRUCTIONS CONDITIONNELLES

A) Qu'est-ce qu'une instruction conditionnelle ?

Définition : Une **instruction conditionnelle** est une instruction qui dépend d'une condition. Certaines parties du code sont exécutées ou ignorées selon que la condition est vérifiée ou non.

Exemple n°1 : On considère l'algorithme suivant.

A ← 27
Si A est pair, alors :
  A ← A ÷ 2
Sinon :
  A ← 3 × A + 1
Fin Si
	La variable A prend la valeur 27.
	On teste la condition "A est pair".
	Si elle est vraie, on applique le code de cette ligne.
	Si la condition est fausse,
	on applique la condition de cette ligne.
	Le bloc d'instruction se termine ici ; tout code qui suivra sera exécuté, indépendamment de la condition testée.

À la fin de l'algorithme, la valeur de A est 82.

B) Utilisation du SI

Dans les instructions conditionnelles, il est possible de n'exécuter un code que si la condition est vérifiée sans donner de code de substitution dans le cas où la condition n'est pas vérifiée. Lorsque cela est nécessaire, on peut alors utiliser SI sans utiliser SINON. Comme d'habitude, tout ce qui suit le bloc d'instructions conditionnelles est exécuté.

Exemple n°2 : L'algorithme suivant utilise également la définition d'une fonction. Ne pas hésiter à se référer à la fiche n°3 en cas de besoin.

```
Fonction EstPaire(Nombre):
    Pair ← Faux
    Si le reste de la division euclidienne de Nombre par 2 est 0, alors :
        Pair ← Vrai
    Fin Si
    Retourner Pair
Fin Fonction
Afficher EstPaire(27)
Afficher EstPaire(28)
```

Détaillons l'exécution de cet algorithme pour l'affichage de EstPaire(27) :
EstPaire(27) signifie que l'on exécute la fonction EstPaire() avec l'argument 27. La variable Nombre prend donc la valeur 27. La variable Pair prend la valeur Faux dès l'appel à la fonction. Ensuite, on teste la condition "le reste de la division euclidienne de Nombre par 2 est 0". Ici, la condition n'est pas vérifiée car le reste est égal à 1. Par conséquent, le code "Pair ← Vrai" n'est pas exécuté.
La ligne "Fin Si" indique la fin du bloc d'instructions conditionnelles. Tout ce qui suit est donc exécuté, à savoir "Retourner Pair". La fonction sauvegarde alors la valeur de la variable Pair (qui est donc "Faux"). Finalement, l'affichage sera "Faux".

C) Utilisation du SI - SINON

Comme dans l'exemple n°1, il arrive (régulièrement) d'utiliser une instruction SI avec une instruction SINON. Cela permet d'exécuter un code spécifique même si la condition testée n'est pas vérifiée.

Exemple n°3 : On considère l'algorithme suivant.

```
Fonction PlusGrand(x, y):
    Si x > y, alors :
        Retourner x
    Sinon :
        Retourner y
    Fin Si
Fin Fonction
```

Dans cet exemple, on teste la condition "x > y" où x et y sont les deux arguments de la fonction PlusGrand().
Si cette condition est vérifiée, alors la valeur retournée est x.
Si la condition est fausse, autrement dit si y ? x, alors, la valeur retournée est y.
En particulier, si x = y, alors la valeur retournée est y.

On peut alors préciser l'algorithme en utilisant plusieurs instructions conditionnelles imbriquées. Dans ce cas, pour des raisons de lisibilité dans un algorithme, il est utile de bien utiliser les alinéas pour montrer les différents niveaux d'imbrication et le début et la fin de chaque bloc d'instructions.
Avec Python, l'utilisation des alinéas est obligatoire (voir partie **D**).

Exemple n°4 :

```
Fonction PlusGrand(x, y):
   Si x = y, alors :
      Retourner “Les valeurs sont égales”
   Sinon :
      Si x > y, alors :
         Retourner x
      Sinon
         Retourner y
      Fin Si
   Fin Si
Fin Fonction
```

Dans cet exemple, on commence par tester la condition “x = y”. Si elle est vraie, la fonction retourne une chaîne de caractères et, dans ce cas, les instructions écrites après le “Sinon” ne sont donc pas exécutées.
Si la condition “x = y” est fausse, on exécute les instructions indiquées après le “Sinon”. Ce bloc commence par un autre blocs d’instructions conditionnelles. Donc, on teste maintenant si la condition “x > y” est vérifiée. Si elle est vraie, on exécute “Retourner x” et l’algorithme s’arrête. Si elle est fausse, on exécute l’instruction après le “Sinon” et on exécute “Retourner y”.

D) Programmer des instructions conditionnelles avec Python

Symboles de comparaison :
== : ce symbole teste si deux valeurs sont égales.
!= : ce symbole teste si deux valeurs sont différentes.
> : ce symbole teste si la première valeur est strictement supérieure à la deuxième.
>= : ce symbole teste si la première valeur est supérieure ou égale à la deuxième.
< : ce symbole teste si la première valeur est strictement inférieure à la deuxième.
<= : ce symbole teste si la première valeur est inférieure ou égale à la deuxième.

On donne ici la programmation sous Python des exemples précédents :

Exemple n°2

```
1  def EstPaire(Nombre):
2    Pair = False
3    if Nombre % 2 == 0:
4      Pair = True
5    return Pair
6
7  print(EstPaire(27))
8  print(EstPaire(28))
```

Quelques remarques sur le code de l'exemple 2 :
Ligne 1 : La définition d'une fonction a été vue dans la fiche n°3. Ici, on définit la fonction EstPaire() qui prend un argument.
Ligne 2 : La variable Pair est définie et on lui assigne la valeur False.
Ligne 3 :

- "if" est l'instruction pour "si" ;
- Le symbole % permet de calculer le reste d'une division euclidienne ;
- Le **symbole ==** est utilisé pour tester si la valeur de gauche (Nombre % 2) est égale à la valeur de droite (0). On **ne peut pas** utiliser le symbole = qui est réservé aux déclarations de variables ;
- Les deux-points en fin de ligne sont obligatoires.

Ligne 4 : L'indentation permet d'indiquer que l'on se trouve à l'intérieur d'un bloc d'instructions conditionnelles.
Ligne 5 : Il n'y a plus d'indentation, ce qui indique que l'on est sorti du bloc d'instructions conditionnelles. En revanche, on est toujours dans le bloc de définition de la fonction. Cette ligne sera toujours exécutée par le programme indépendamment du résultat du test de la ligne 3.

Exemple n°3

```
1  def PlusGrand(x, y):
2    if x > y:
3      return x
4    else:
5      return y
```

Quelques remarques sur le code de l'exemple 3 :
Lignes 2 et 3 : Le bloc d'instruction commence avec un "if" et les deux-points sont bien indiqués à la fin de la ligne. Si la condition est vérifiée, alors la ligne 3 (qui est indentée) est exécutée.

Lignes 4 et 5 : “else” signifie “sinon”. Le “else” doit être au même niveau d’indentation que le “if” pour montrer que l’on est toujours dans le même bloc conditionnel. Le deux-points est encore une fois obligatoire. La ligne 5 est indentée. Les lignes 4 et 5 ne seront exécutées que si la condition de la ligne 2 est fausse. Si la condition est vraie, les lignes 4 et 5 sont ignorées.

Exemple n°4

```
1  def PlusGrand(x, y):
2    if x == y:
3      return "Les deux valeurs sont égales"
4    else:
5      if x > y:
6        return x
7      else:
8        return y
```

Quelques remarques sur le code de l’exemple 4 :
Lignes 5 à 8 : Ces lignes ne sont exécutées que si la condition de la ligne 2 est fausse. Dans ce cas, le programme continue au “else” de la ligne 4 qui a le même niveau d’indentation que le “if” de la ligne 2.
Par ailleurs, ce bloc conditionnel des lignes 5 à 8 est identique à celui de l’exemple n°3 mais il a subi plusieurs indentations pour bien être interprété à l’intérieur de l’instruction “else” de la ligne 4.

Comment utiliser « elif » ?
L’instruction “elif” peut être utilisée en raccourci d’une instruction “else” suivi d’une instruction “if”. Utilisée convenablement, l’instruction “elif” est très pratique lorsqu’on utilise plusieurs boucles conditionnelles emboîtées. Le code ci-dessous est équivalent au code précédent.

```
1  def PlusGrand(x, y):
2    if x == y:
3      return "Les deux valeurs sont égales"
4    elif x > y:
5      return x
6    else:
7      return y
```

L’instruction “elif” de la ligne 4 n’est exécutée que si la condition testée à la ligne 2 est fausse.

E) Exercices

✓ **Exercice 1 :**

1) Écrire un algorithme qui teste si un nombre donné en argument d'une fonction est un multiple de 3.
2) Programmer cet algorithme avec Python.

✓ **Exercice 2 :**

Avec Python, le test "Condition1 and Condition2" est vrai lorsque les deux conditions Condition1 et Condition2 sont vraies. Par exemple, le test suivant est vrai :

```
(1 < 2) and (6 > 5)
```

Déterminer la valeur (Vrai ou Faux) de chacun des tests suivants.

```
1   (1 < 2) and (6 < 5)
2   (4 == 2 + 2) and (6 != 7)
3   (3 <= 2 + 1) and (2 > 3)
```

✓ **Exercice 3 :**

On a programmé avec Python une fonction MultipleDeux(Nombre) qui retourne Vrai lorsque la variable Nombre est un multiple de 2 et Faux dans le cas contraire.

1) Programmer une fonction MultipleCinq(Nombre) qui retourne Vrai lorsque la variable Nombre est un multiple de 5 et Faux dans le cas contraire.
2) Quel est le but de la fonction programmée ci-dessous ? Justifier.

```
13  def Mystere(Nombre):
14      return MultipleDeux(Nombre) and MultipleCinq(Nombre)
```

✓ **Exercice 4 :**

Programmer une fonction avec Python qui prend deux arguments correspondant respectivement à la longueur et la largeur d'un rectangle. Cette fonction doit renvoyer la valeur True si le rectangle est un carré et False sinon.

✓ **Exercice 5 (nécessite le chapitre sur les équations de droite) :**

On considère deux points A et B de coordonnées respectives $(x_A\,;\,y_A)$ et $(x_B\,;\,y_B)$ dans un repère orthonormé.
1) Programmer une fonction avec Python qui prend en arguments les deux coordonnées de chacun des deux points (soit quatre arguments en tout) et qui renvoie la valeur Vrai lorsque les deux points ont la même abscisse et Faux sinon.
2) Utiliser la fonction alors définie pour que le programme précise la forme de l'équation réduite de la droite (AB) ($x = k$ ou $y = mx + p$).