Tableau de
Déclaration des objets (TDO)

Objet	Type / Nature
Nom_tableau	Nom type

3ème * Sciences Informatiques

ALGORITHMIQUE & PROGRAMMATION

Leçon 5

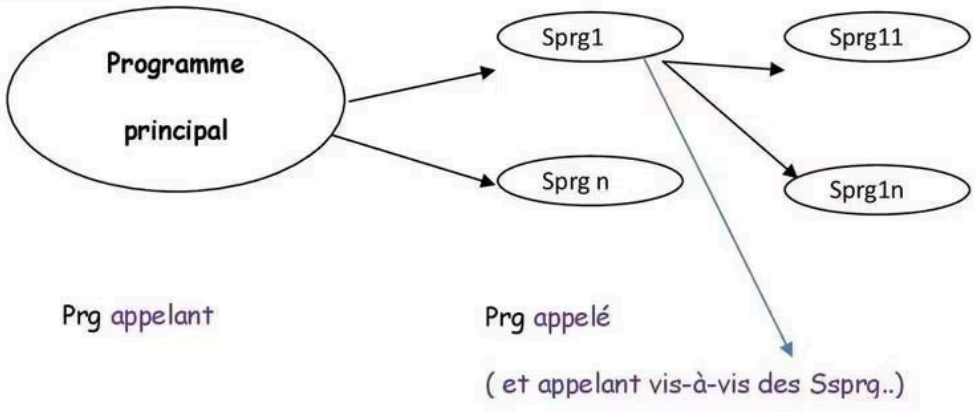
Les sous programmes

I. Introduction

1-Analyse modulaire

L'analyse modulaire consiste à diviser un problème en sous problèmes de difficultés moindres. Ces derniers sont aussi soumis à cette division jusqu'à ce qu'on arrive à un niveau abordable de difficulté.

- ✓ Un sous-programme est un ensemble d'instructions, déclaré dans un programme ou dans un sous-programme.
- ✓ Il peut être une fonction ou une procédure.
- ✓ Un sous-programme peut être exécuté plusieurs fois grâce à des appels (il permet d'éviter la redondance des programmes).



2- Intérêts

- ✓ Présenter des solutions claires en évitant la redondance des codes dans un programme
- ✓ Simplifier la résolution du problème initial en supposant que les différents modules prévus sont déjà implémentés
- ✓ Se concentrer sur la résolution d'un sous problème à la fois.
- ✓ Détecter facilement les parties à consulter ou à modifier.

II. Les fonctions

Une fonction est un sous-programme qui retourne un seul résultat de type simple (entier, réel, booléen, caractère ou chaine de caractères).

Syntaxe d'appel d'une fonction En Algorithme:

```
Variable ← nom-fonction (paramètres effectifs)        # affectation
Ecrire (nom-fonction (paramètres effectifs))          # Affichage
Si (nom-fonction (paramètres effectifs)) = valeur) alors ...   # Condition
```

Syntaxe de la définition (déclaration) d'une fonction

➢ En algorithmique une fonction est déclarée en utilisant la syntaxe suivante :

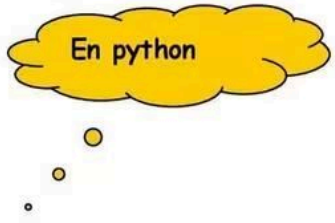
```
Fonction nom-fonction (parf1 :type1, parf2 :type2,..,parfn :typen) : type du résultat

Début
    Instruction 1
    Instruction 2
    ...
    Instruction n
    Retourner Resultat_calculé
Fin
```

➢ En Python une fonction est déclarée en utilisant la syntaxe suivante :

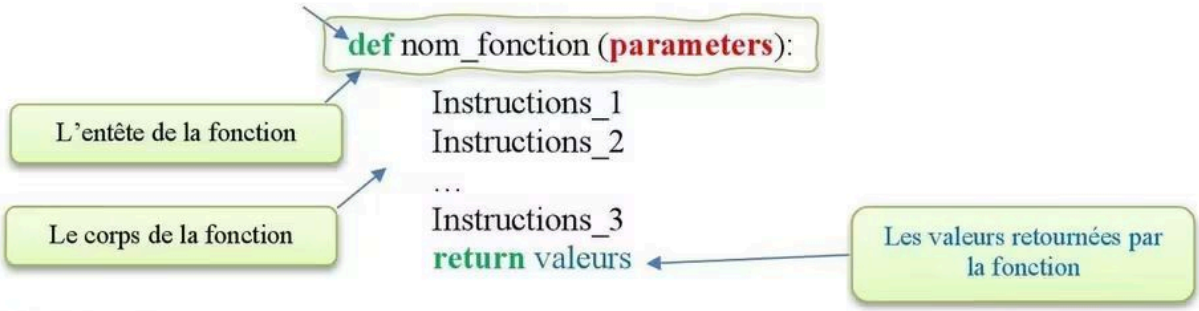
```
def nom-fonction (parf1, parf2,..,parfn) :

    Instruction 1
    Instruction 2
    ...
    Instruction n
    return Resultat
```



Définition de la fonction

Paramètres formels :
Les objets nécessaires pour que la fonction puisse faire sa tâche (son travail).



Remarques

1- Pour exécuter une fonction, il faut **l'appeler**.
2- Toutes les instructions qui suivent l'instruction **return** constitue un code mort c'est-à-dire non exécutable par la fonction.
3- Une fonction peut ne pas avoir de paramètres au moment de la définition, dans ce cas il ne faut pas oublier les deux parenthèses ().
4- Une fonction peut renvoyer une ou plusieurs valeurs de types différents dans la même instruction return.

Objets globaux et objet locaux

Un Objet Global est tout objet déclaré dans le programme principal (PP), il est utilisable par les instructions du PP et par les instructions des différents sous-programmes
Un Objet Local est tout objet déclaré à l'intérieur d'un sous-programme.
Un objet local ne peut pas être utilisé en dehors du sous-programme où il est déclaré

Paramètres effectifs et paramètres formels

Les paramètres formels sont des variables utilisées dans le corps du sous-programme, ils figurent dans l'entête de la déclaration d'un sous-programme. Les paramètres formels sont des objets locaux
Les paramètres effectifs sont des variables (ou valeurs) fournies lors de l'appel d'un sous-programme, ils figurent dans l'instruction d'appel du sous-programme, au niveau du programme appelant (Programme principal). Les paramètres effectifs sont des objets globaux

Remarques

- Il faut respecter le nombre, l'ordre et les types des paramètres effectifs par rapport aux paramètres formels. Leurs noms peuvent être différents.
- Au moment de l'appel, les paramètres formels seront remplacés par les paramètres effectifs lors de l'exécution du sous-programme

III. Les procédures

Une procédure est un sous-programme (ensembles d'instructions) qui peut produire Zéro ou plusieurs résultats. Dans une procédure, il faut ajouter le symbôle @ avant chaque objet résultat.

➢ Syntaxe de la définition (déclaration) d'une procédure

➢ En algorithmique une procédure est déclarée en utilisant la syntaxe suivante :

```
Procédure nom-Procédure (liste des paramètres formels : type de chaque paramètre)
Début
    Instruction 1
    Instruction 2
    ...
    Instruction n
Fin
```

Syntaxe de l'appel d'une procédure

L'appel de la procédure se fait en utilisant simplement le nom de la procédure suivi d'une liste de paramètres (les paramètres effectifs).

En algorithme

Nom-procédure (paramètres effectifs)

➢ Mode de passage

Le passage de paramètres par valeur

Dans ce mode de passage, le programme appelant (principal) transmet une **valeur** au programme appelé (procédure). Toute modification de la valeur de ce paramètre par le programme appelé ne sera pas transmise au programme appelant.

Le passage de paramètres par adresse

Dans ce mode de passage le programme appelant (principal) et le programme appelé (procédure) font un échange de données. En effet tout changement de la valeur d'un paramètre formel sera transmis au paramètre effectif correspondant pendant et après l'exécution du programme appelé.

Dans le mode passage par adresse, on fait précéder les paramètres formels par le symbole @, dans l'entête de la procédure.

Exemple1 : Mode de passage par adresse

Dans le programme principal l'appel de la procédure saisir se fait comme suit :

```
Algorithme exemple
Début
 saisir (a , b)
Fin
```

Algorithme de la procédure Saisir

```
Procédure Saisir ( @ a1 : entier ; @ b1 : entier)
Début
 Répéter
     Ecrire ("Donner un entier : "), Lire (a1)
  Jusqu'à (a1>=1)
  Répéter
     Ecrire ("Donner un entier : "), Lire (b1)
```

Jusqu'à (b1>=a1)
Fin

NB : a1 et b1 contiendront les 2 valeurs résultats de la procédure Saisir, pour le mentionner, nous devons les faire précéder dans l'entête par le symbole @, nous parlons alors de **mode de passage** des paramètres par **adresse** (ou par **référence**)

Exemple2 : Mode de passage par valeur

```
Algorithme principal
Algorithme exemple
Début
  Lire (a)
  Lire (b)
  P1 (a , b)
Fin
```

```
Algorithme de la procédure P1
Procédure P1 ( x, y : Entier)

Début
  Traitement
Fin
```

Au moment de l'appel de la procédure **P1** dans le programme principal, les paramètres effectifs (a et b) passeront leurs valeurs aux paramètres formels (x et y).

Conclusion

- **Dans une fonction, on n'a que le mode de passage par valeur**
- **Dans une procédure, on peut avoir le mode de passage par valeur et par adresse**

3ème * Sciences Informatiques

Leçon 6