Les sous-programmes : Fonctions et Procédures

Pourquoi les sous-programmes ?

Dans un algorithme (ou programme informatique) on se retrouve souvent avec des bouts de codes qui peuvent se répéter dans plusieurs emplacements. Cette redondance rend le code long, moins lisible et, par-suite, si on souhaite modifier le comportement de ces blocs redondants alors on doit les réécrire tous, ce qui n'est pas du tout pratique.

Les sous-programmes permettent d'éviter cette redondance en factorisant le code qui peut se répéter. De cette façon nous n'aurons qu'un seul bloc de code que l'on peut réutiliser à volonté dans plusieurs endroits sans être obligé de le réécrire.

Les sous-programmes peuvent être des **fonctions** ou des **procédures**. Les fonctions et les procédures sont presque similaires, mais ***une fonction retourne un et un seul résultat de type simple***, alors qu'***une procédure ne retourne rien***.

I. Les Fonctions

1. Déclaration d'une fonction en algorithme et en Python :

Algorithme	Python

Fonction Nom_fonction (pf1 : type, pf2 : type, …) : type_résultat
Début
Traitement
Retourner *resultat*
Fin
	`def Nom_fonction (pf1,pf2,…):`
	`  Traitement`
	`  return resultat`

2. Appel d'une fonction en algorithme et en Python (au niveau du programme principal) :

L'appel d'une fonction, en algorithme, comme en python, peut se faire de deux méthodes :

Appel algorithmique	Appel en Python

Objet←Nom_fonction (pe1, pe2, …, pen)
Ou bien
Ecrire (Nom_fonction (pe1, pe2, …, pen))
	`Objet=Nom_fonction(pe1,pe2,…,pen)`
	Ou bien
	`print(Nom_fonction(pe1,pe2,…,pen))`

NB :

- *pf :* est un **paramètre formel** qui est défini au niveau de la définition de la fonction.
- *pe :* est un **paramètre effectif** utilisé lors de l'appel de la fonction.
- L'instruction **return**, définit ce que doit renvoyer la fonction.

Application 1 : écrire un algorithme et son implémentation en Python d'une fonction **Carre(x)** qui permet de calculer le carré d'un entier "x", ainsi que son programme principal.

Correction :

En algorithme

```
Algorithme calcul_carre
    Fonction Carre(nbr : entier) : entier
    Début
        retourner (nbr*nbr)
    Fin
Début (programme principal)
    Écrire ("taper un entier : ")
    Lire(x)
    Écrire (x,"²=",Carre(x))
Fin.
```

TDOL : (pas d'objets locaux à déclarer)

O	T

TDOG :

O	T
x Carre	Entier Fonction

En Python

```
1 # fonction Carre
2 def Carre(nbr):
3     return(nbr*nbr)
4 #programme principal
5 x=int(input("taper un entier: "))
6 print(x,"²=",Carre(x))
```

```
Console
taper un entier: 4
4 ²= 16
```

Application 2 : écrire un algorithme et son implémentation en Python d'une fonction **NbVoy(ch)** qui permet de retourner le nombre de voyelles dans une chaine de caractères. Exp : ch= "Salut mes amis" ➔ nb = 5.

Correction :

En algorithme	En Python

```
Algorithme calcul_voyelles
    Fonction NbVoy(ch:chaine de caractères):entier

    Début
        nb←0
        Pour i de 0 à long (ch)-1 faire
            Si(Majus(ch[i]) ∈ ["A","E","I","O","U","Y"] ) alors
                nb←nb+1
            Fin si
        Fin pour
        Retourner nb
    Fin
Début #programme principal
    Écrire("taper une chaine de caractère : ") Lire(ch)
    Écrire("cette chaine contient ",NbrVoy(ch),"voyelles")
Fin.
```

TDOL :

O	T
nb, i	entier

TDOG :

O	T
Ch	Chaine de caractères
NbVoy	Fonction

```python
def NbrVoy(ch):
    nb=0
    for i in range(len(ch)):
        if(ch[i].upper()in["A","E","I","O","U","Y"]):
            nb=nb+1#nb+=1
    return(nb)
ch=input("taper une chaine de caractère : ")
print("cette chaine contient",NbrVoy(ch),"voyelles")
```

```
>>> %Run fonction_nbr_voy.py
taper une chaine de caractère : I love coding with python!
cette chaine contient 8 voyelles
```

Application 3 : écrire un code, en algorithme et en Python, d'une fonction **Nbchiff(ch)** qui permet de retourner le nombre de chiffres dans une chaine de caractères. Exp : ch= "Bac2022" nb = 4.

Correction :

En algorithme	En Python

```
Algorithme calcul_chiffres
    Fonction Nbchiff(ch:chaine de caractères):entier

    Début
        nb←0
        Pour i de 0 à long (ch)-1 faire
            Si "0"<=ch[i]<="9" alors
                nb←nb+1
            Fin si
        Fin pour
        Retourner nb
    Fin
Début #programme principal
    Écrire("taper une chaine de caractère : ") Lire(ch)
    Écrire(ch,"contient ",Nbchiff(ch),"chiffres")
Fin.
```

TDOL :

O	T
nb, i	entier

TDOG :

O	T
Ch	Chaine de caractères
NbVoy	Fonction

```python
def Nbchiff(ch):
    nb=0
    for i in range(len(ch)):
        if "0"<=ch[i]<="9":
            nb=nb+1
    return nb
def NbrVoy(ch):
    n=0
    for i in range(len(ch)):
        if(ch[i].upper()in["A","E","I","O","U","Y"]):
            n=n+1
    return(n)
#programme principal
ch=''
while not(len(ch)>1):
    ch=input('taper une chaîne non vide ')
print(ch,"contient",Nbchiff(ch),'chiffres')
print("cette chaine contient",NbrVoy(ch),"voyelles")
```

II. Les Procédures :

1. Déclaration d'une procédure en algorithme et en Python :

Algorithme	Python
Procédure Nom_procédure (pf1 : type1, pf2 : type2, …) Début Traitement Fin	**def** Nom_procédure (pf1, pf2,…) Traitement

2. Appel d'une procédure :

L'appel d'une procédure, au niveau Algorithme, comme en python se fait par son nom suivi par la liste de paramètres effectifs, comme suit :

En algorithme	En Python
Nom_Procédure (pe1,pe2, …)	Nom_Procédure (pe1,pe2, …)

Application 1 : écrire un code, en algorithme et en Python, d'une procédure **lecture (ch)** qui permet de saisir une chaine de caractères non vide et de longueur maximale =20.

Correction :

En algorithme

```
Algorithme saisie_chaine
Procédure lecture(@ ch : chaîne de caractères)
    Début
        Répéter
            Écrire("taper une chaîne de longueur max 20")
            Lire(ch)
        Jusqu'à (0<long(ch)<=20)
    Fin
Début
    lecture(mot)
    écrire("vous avez tapé",mot)
Fin.
```

TDOL :

Objets	Type/Nature

TDOG :

Objets	Type/Nature
mot	Chaîne de caractères
lecture	procédure

En Python

```
#procédure lecture:
def lecture():
    ch=""
    while not(0<len(ch)<=20):
        ch=input("taper une chaine non vide ")
    return ch
#programme principal:
mot=lecture()
print("vous avez tapé",mot)
```

```
Console
   %Run saisie_ch.py
taper une chaine non vide bonjour
vous avez tapé bonjour
```

Exercice : écrire un algorithme(+TDO nécessaires) et son implémentation en python qui :

- Saisit 2 entiers n et m compris entre 1 et 10 ;
- Calcule et affiche n^m (n à la puissance m).

NB : une solution modulaire est exigée.

```
Algorithme exercice
    Procédure Saisie_a(@ a : entier)
        Début
            Répéter
                Écrire("donner un entier : ")
                Lire(a)
            Jusqu'à a ∈ [1..10]
        Fin
    Fonction puissance(x : entier, y : entier) : entier
        Début
            p←1
            Pour i de 1 à y faire
                p←p*x
            fin pour
            Retourner p
        Fin

Début
    Saisie_a(n)
    Saisie_a(m)
    Écrire(n,"à la puissance",m,"=",puissance(n,m))
Fin.
```

TDOL :

O	T

TDOL :

O	T
p	entier
i	entier, compteur

TDOG :

O	T
n,m	entier
Saisie_a	procédure
puissance	fonction

```
#procédure de saisie:
def Saisie_a():
    a=0
    while not(1<=a<=10):
        a=int(input("donner un entier : "))
    return a
#fonction de calcul puissance:
def puissance(x,y):
    p=1
    for i in range(y):
        p=p*x
    return p
#programme principal:
n=Saisie_a()
m=Saisie_a()
print(n,"à la puissance",m,"=",puissance(n,m))
```

Application 2 : écrire un code, en algorithme et en Python, d'une procédure remplir(n,T) qui permet de remplir un tableau T de n caractères alphabétiques.

Application 3 : écrire un algorithme modulaire « traitement » qui permet de :

- Saisir un entier N avec (3 <= N <= 30)
- Remplir un tableau par N caractères alphanumériques.
- Déterminer le nombre de lettres se trouvant dans ce tableau T
- Déterminer la chaine formée par les chiffres existant dans le tableau.
- Afficher le tableau T ainsi que le nombre de lettres et la chaine des chiffres.

Exemple :

Si N= 8 et T :

2	M	0	A	2	T	2	H
0	1	2	3	4	5	6	7

Alors le programme affichera : 2 M 0 A 2 T 2 H

Le nombre de lettres = 4

La chaine des chiffres = 2022