Algorithmique et Programmation 1

Fonctions

Licences Informatique et Mathématiques
1ère année

Premier semestre 2015 – 2016

Fonction ? pour quoi faire ?

- Lisibilité - Pour séparer une partie du programme...

  par exemple un gros calcul compliqué
- Modularité - ... et pouvoir la réutiliser ...

  évite de recopier le même code à plusieurs endroits
- Généricité - ... en changeant la valeur de ses paramètres

  même calcul mais avec différentes valeurs de départ

Vous utilisez déjà des fonctions dans vos programmes :

```
nb = input("donnez un nombre")
tirage = randint(1, 6)
print("blabla")
```

```
x = min(12, x)
str(3.14)
ligne(0, 50, 200, 50)
```

Syntaxe de la définition de fonction

```
# ligne suivante : en-tête de fonction
def nom_fonction(param_1, ..., param_n):
    # bloc d'instructions indentées
    # (appelé corps de la fonction)
    # utilisant param_1 à param_n
    ...
    # peut renvoyer un résultat :
    return resultat
```

Exemple :

```
def maximum(a, b):
    if a > b:
        return a
    else:
        return b
```

Une fonction peut :

- prendre un certain nombre de paramètres (ici, n)
- renvoyer une valeur (via l'instruction `return`)

Une fois définie, `nom_fonction` peut être utilisée dans le code

Appel de fonction

```
def nom_fonction(param_1, ..., param_n):
    # corps de la fonction
    ...

# reste du programme
...
# appel de la fonction :
nom_fonction(val_1, ..., val_n)
# ou pour récupérer la valeur renvoyée :
res = nom_fonction(val_1, ..., val_n)
```

```
def maximum(a, b):
    ...

nb = input("combien?")
scre = maximum(nb, 5)
print("En voici", scre)
```

- val_1, ..., val_n sont des valeurs ou des expressions
  Utilisant uniquement les variables du programme principal (à suivre)
- Si la fonction renvoie une valeur, ne pas oublier de la récupérer
- Les variables de la fonction ne sont pas visibles dans le reste du programme (à suivre)

Fonctions sans paramètre

- Une fonction peut n'avoir aucun paramètre :

```
def lance_de() :
    return randint(1,6)

compteur = 0
while lance_de() != 6:
    compteur = compteur + 1
print('Obtenu un 6 en', compteur, 'jets de dé.')
```

Fonctions sans valeur de retour

- Une fonction peut ne rien renvoyer (pas de `return` ou `return` non suivi d'une expression)

```
def trace_polygone(nb_cotes, taille_cote):
    down()
    i = 0
    while i < nb_cotes:
        forward(taille_cote)
        left(360 / nb_cotes)
        i = i + 1

# faire une affectation ici ne servirait à rien !
trace_polygone(4, 100)
```

Erreurs fréquentes

- Ne pas confondre paramètres et saisies !
- Ne pas confondre sortie et affichage !

```
def pgcd():
    a = int(input())
    b = int(input())
    while a % b != 0:
        r = a % b
        a = b
        b = r
    print("le pgcd est", b)
```

Attention : NE PAS PROGRAMMER COMME ÇA !

- Comment écrire un programme vérifiant si trois entiers sont premiers entre eux à l'aide de cette fonction ?
- Combien ce programme ferait-il de saisies ?
- Qu'afficherait ce programme ?

Composition de fonctions

Composition : appel d'une fonction depuis une autre fonction

```
def pgcd(a, b):
    while a % b != 0:
        r = a % b
        a = b
        b = r
    return b

def premiers_entre_eux(a, b):
    return pgcd(a, b) == 1

# programme principal :
m = int(input())
n = int(input())
o = int(input())
if (premiers_entre_eux(m, n)
    and premiers_entre_eux(n, o)
    and premiers_entre_eux(o, m)):
    print("tous premiers entre eux")
```

Remarque – appels de fonctions

- Un appel de fonction peut être utilisé dans une expression :

```
if max(1, x + y) == 1:
    print('somme de x et y plus grande que 1')
print('le maximum de' x, y, 'et', z, 'est',
      max(x, max(y, z)))
```