Fonctions - Procédures en Python

Structure d'un programme Python

Importations

Fonctions-Procédures

Programme principal

Intérêts:

- Eviter les redondances d'instructions
- Ecriture du code plus rapide
- Réutiliser du code sans avoir à le dupliquer
- Simplifier le code

Il existe deux types de "fonctions":
celles qui renvoie un ou plusieurs résultats (fonctions) et celles qui ne renvoient rien (procédures).

Dans l'entête de la fonction, on peut lui mettre des paramètres qui seront utilisés à l'intérieur.

"fonction" ne renvoyant pas de résultat
(procédure)

```
def nom_de_la_fonction(paramètres) :
    global variables globales*
    # instructions
```

"fonction" renvoyant des résultats
(fonction)

```
def nom_de_la_fonction(paramètres) :
    global variables globales*
    # instructions
    return résultats
```

* on ajoute cette ligne si la fonction doit utiliser ou modifier des variables globales

Variables globales= variables du programme principal

Procédure sans paramètre

```
def carre(): (1)
    for i in range(4):
        joe.forward(100)
        joe.right(90)
```

Appel: carre()
Seule la tortue joe trace un carré.

Procédure ayant en paramètre une tortue t

```
def carre(t): (2)
    for i in range(4):
        t.forward(100)
        t.right(90)
```

Appel: carre(nom_de_la_tortue)
carre(joe)
carre(jack)
joe et jack tracent un carré

Procédure ayant en paramètre une tortue t et une longueur c

```
def carre(t,c): (3)
    for i in range(4):
        t.forward(c)
        t.right(90)
```

Appel: carre(nom_de_la_tortue, longueur)
carre(joe,50), carre(jack,20)
joe trace un carré de côté 50
et jack trace un carré de côté 20.

Fonction sans paramètre

```
def alea():
    return int(random.random()*11)
```

Appel: n=alea()
alea() renvoie un entier n aléatoire de [0;10]

Fonction ayant en paramètre une longueur c

```
def prop_carre(c):
    p=4*c
    a=c*c
    return p,a
```

Appel: peri,aire=prop_carre(longueur)
peri contiendra la valeur de p et aire celle de a.
Attention à l'ordre!

Fonction ayant en paramètre une tortue t et une longueur c

```
def carre(t,c):
    for i in range(4):
        t.forward(c)
        t.right(90)
    p=4*c
    a=c*c
    return p,a
```

Appel: peri,aire= carre(nom_de_la_tortue, longueur)

Dans un programme Python, on ne peut pas déclarer plusieurs fonctions ou procédures de même nom. Pour regrouper les 3 procédures carre en une seule, on définit une tortue et une longueur par défaut.

```
def
carre(tortue=joe,longueur=100):
  for i in range(4):
      tortue.forward(longueur)
      tortue.right(90)
```

Appels:

Procédure **(1)** <=> carre()
Procédure **(2)** <=> carre(jack)
Procédure **(3)** <=> carre(jack, 20)
Procédure **(3)** <=> carre(joe, 50)
<=> carre(***longueur***=50)

En abscence du premier paramètre, il faut indiquer le nom du paramètre longueur.

Les *paramètres ayant des valeurs* par défaut sont **optionnels**.

```
def carre(tortue,longueur):
  for i in range(4):
      tortue.forward(longueur)
      tortue.right(90)
```

Appel: carre(joe, 50)

Les paramètres sont **indispensables**.

Cf tester le programme Fonction_Ex1.py du dossier Fonctions_Procedures dans le WORKSPACE

Défi A01a
Créer une procédure *triangle* ayant pour paramètres une tortue, une longueur

Défi A01b
Créer une procédure *triangle* ayant pour paramètres une tortue t, une longueur c , une couleur de trait pc.
La longueur par défaut est 50, la couleur par défaut est vert.

Défi A01c
Créer une procédure *triangle* ayant pour paramètres une tortue, une longueur, une couleur de trait.
La longueur par défaut est 50, la couleur par défaut est None.

Défi A02a
Créer une procédure *hexagone* ayant pour paramètres une tortue, une longueur, la couleur du trait, la largeur du trait.
Valeurs par défaut:
longueur=40
couleur='red'
largeur=5

Défi A02b
Idem défi A02a mais avec un *pentagone* .

Défi A02c
Idem défi A02a mais avec un *octogone*

Défi A03a
Idem défi A02a mais avec un paramètre supplémentaire: un booléen b qui vaut True si on colorie l'intérieur de la figure.

Défi A03b
Idem défi A03a mais avec un *pentagone* .

Défi A03c
Idem défi A03a mais avec un *octogone*