07 - Introduction au module Turtle

November 29, 2023

1 Découverte de la bibliothèque Turtle, un module de dessin

1.0.1 Principe général

Vous déplacez une tortue sur une feuille de papier. En se déplaçant elle trace un trait (mais on peu lui demander de se déplacer sans tracer)

Tout d'abord découvrons les premières fonctions de base et apprenons à nous repérer sur la feuille

Commandes de base: - `turtle.forward(N)` - fait avancer la tortue de N pixels. - `turtle.done()` - conclut la séquence Turtle.

La Tortue, au départ, est au milieu de la feuille et prête à se déplacer vers la droite

Faisons la avancer de 50 pixels.

[ ]:
```
import  turtle

turtle.forward(50)

# on terminera toujours le code par cette instruction :
turtle.done()
```

- `turtle.backward(N)` - la tortue peut aussi reculer!

Faisons la avancer de 50 pixels puis reculer d'autant

[ ]:
```
import  turtle

turtle.forward(50)
turtle.backward(50)

# on terminera toujours le code par cette instruction :
turtle.done()
```

La Tortue est revenue à sa position initiale….

- `turtle.left(N)`- elle peut aussi tourner de N degrés vers la gauche….
- `turtle.right()`- ou vers la droite.

Faisons la se promener sur 3 segments de droite…

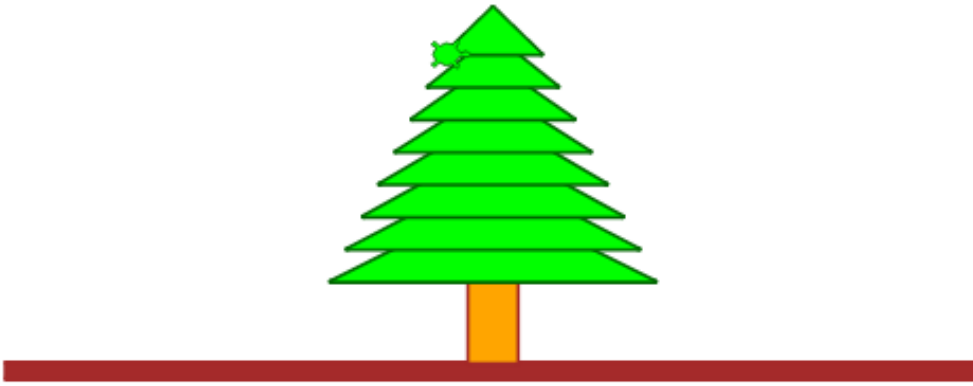
[ ]:
```
import  turtle

turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)


# on terminera toujours le code par cette instruction :
turtle.done()
```

C'est à vous! Dessinez avec la tortue un carré de coté 50 - en utilisant une boucle (le code ne doit pas dépasser 4 lignes).

[ ]:
```
# a vous
```

Une fonction carre: créez une fonction nommée carré qui prend un seul paramètre (un nombre entier *cote*) ***et qui dessine un carré dont les cotés ont une longueur*** *cote*

[ ]:
```
# ecrire la fonction ici
def carre(cote):
    """
    parametre : cote (int) est la taille (en pixels) des cotés du carré
    Sortie: lafonction ne renvoie rien,elledessigne un carré de taille "cote"
    """

############################
#  programme principal   #
# appels de la fonction  #
############################
cote = 50
carre(cote)

cote = 100
carre(cote)

turtle.done()
```



Vous devez avoir obtenu ceci :

Améliorons notre fonction carre

Recopiez la fonction précédente pour qu'elle accepte 3 paramètres + un nombre entier n + un entier *epaisseur* qui détermine l'epaisseur du trait + une variable str *couleur* désignant une couleur

et qui dessine un carré dont les cotés ont une longueur n l'épaisseur du trait est *epaisseur* et la couleur est *couleur*

Aide: + Pour changer l'épaisseur du trait on appelle par exemple:

```
turtle.pensize(3)
```

tout les tracés qu'on fait ensuite auront une épaisseur de 3 pixels

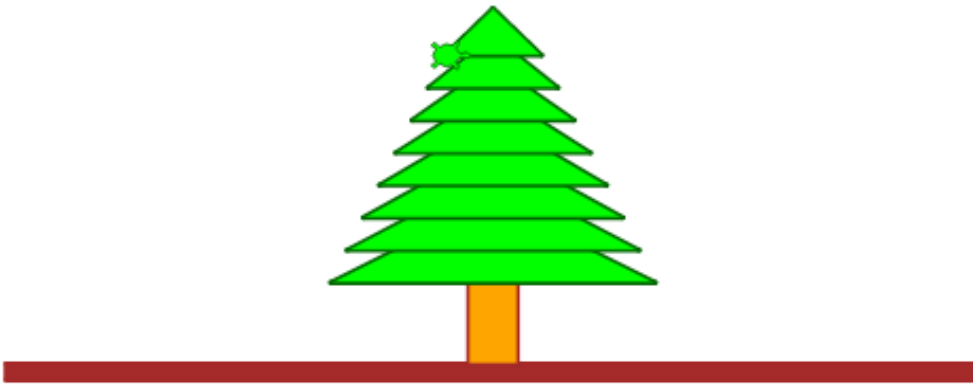
- Pour changer la couleur, on appelle par exemple.

```
turtle.color("red")
```

Les noms des couleurs sont bien sûr en anglais...

[ ]:
```
# a vous



##########################
#  programme principal   #
# appels de la fonction  #
##########################
cote = 50
epaisseur = 3
couleur = "red"
carre(cote, epaisseur, couleur)

cote = 100
epaisseur = 3
couleur = "blue"
carre(cote, epaisseur, couleur)
```

```
turtle.done()
```



Vous devez avoir obtenu ceci :

Déplacer la tortue sans qu'elle dessine: - `t.up()` - soulève la tortue (elle cesse donc de dessiner mais peut encore se déplacer) - `t.down()` - fait l'inverse - `t.goto(x, y)` - envoie directement la tortue au point de coordonnées (x,y) à partir de sa position actuelle, en ligne droite. *Sachant que le point (0,0) est le centre de la fenêtre.*

Le code ci-dessous va utiliser votre fonction carré pour déssiner des carré a plusieurs endoits du dessin:

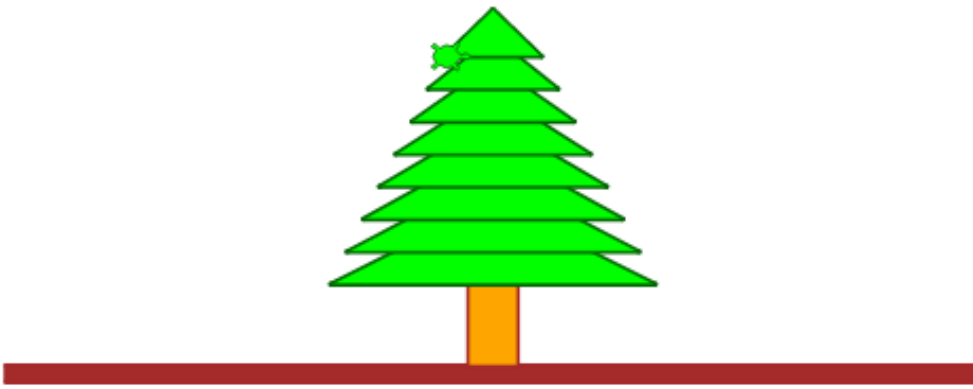
[ ]:
```
x = 100
y = 120

turtle.up()     # on lève lestylo
turtle.goto(x, y) # on déplace la tortue sans tracer
turtle.down()   # on abaisse le stylo
carre(50, 3, "green")


x = -100
y = 120

turtle.up()
turtle.goto(x, y)
turtle.down()
carre(50, 3, "salmon")

for i in range(5) :
    x = -100 + i*50
    y = 150
    turtle.up()
    turtle.goto(x, y)
    turtle.down()
    carre(50, 3, "purple")
```

```
turtle.done()
```

Améliorons encore la fonction carre

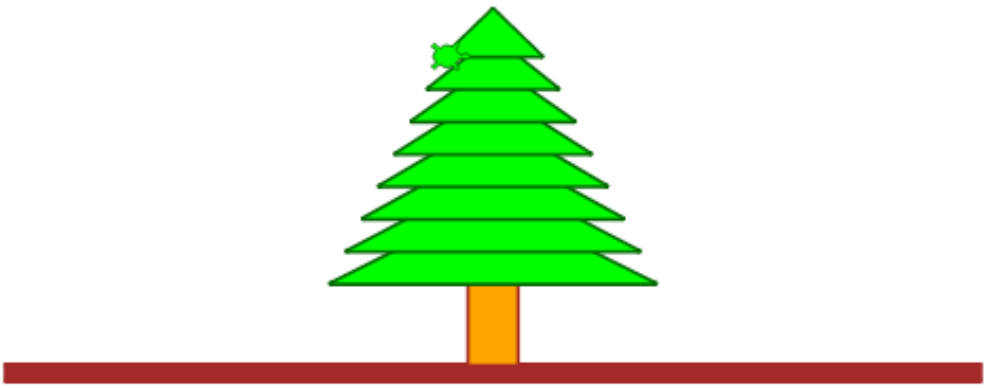
En vous inspirant du code précédent, modifiez la fonction pour qu'elle puisse répondre aux spécifications indiquées ci-dessous (recopiez le code ci-dessus et modifiez-le).

[ ]:
```
def carre(x, y, cote, epaisseur, couleur):
    """
    Parametres :
        x et y (int) : coordonnées du coin inférieur droit de notre carré
        cote (int) : la longuer des cotés du carré
        epaisseur (int) : epaisseur du trait (en pixels)
        couleur (str): le nom d'une couleur (en anglais)
    """
    # complétez...

############################
#  programme principal   #
# appels de la fonction  #
############################
carre(-100, -100, 100,3,"blue")

carre(100, 100, 50,3,"lime")

# affichage du dessin
turtle.done()
```



Vous devez avoir obtenu ceci :

Voici la liste des couleurs précodées avec leurs noms:



Remplir la forme dessinée: on peut remplir le carré en utilisant `turtle.begin_fill()` au début du tracé et `turtle.end_fill()` à la fin.

Par défaut la couleur de remplissage est la même que pour les traits, mais on peut spécifier une couleur de remplissage avec

`turtle.fillcolor("nom de la couleur")`

Attention : cette instruction fillcolor doit être placée APRES

`turtle.color("nom de la couleur")`

sinon color modifiera la couleur de remplissage.

Voici un exemple:

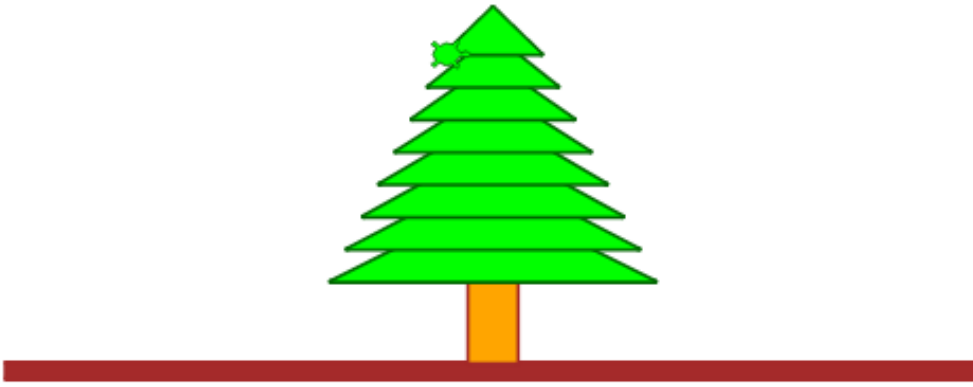
[ ]:
```
import turtle

x= 100
y = 100

# déplacement
turtle.up()
turtle.goto(x, y)
turtle.down()

# paramètre du stylo
turtle.pensize(3)
turtle.color("blue")

# début de remplissage
turtle.begin_fill()
turtle.fillcolor("lightblue")

# dessin du carré
for i in range(4) :
```

```
    turtle.forward(cote)
    turtle.left(90)

# fin du remplissage
turtle.end_fill()



# affichage du dessin
turtle.done()
```

Améliorons encore la fonction carre

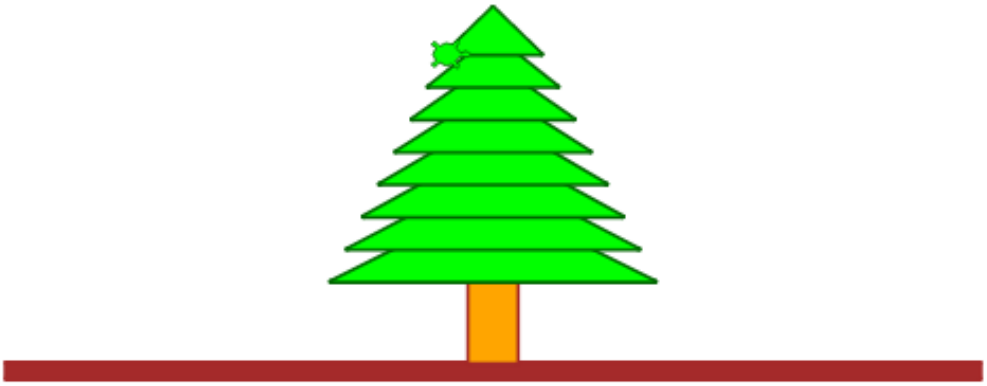
En vous inspirant du code précédent, modifiez la fonction pour qu'elle puisse répondre aux spécifications indiquées ci-dessous

[ ]:
```
def carre(x, y, cote, epaisseur, couleur, remplissage):
    """
    Parametres :
        x et y (int) : coordonnées du coin inférieur droit de notre carré
        cote (int) : la longuer des cotés du carré
        epaisseur (int) : epaisseur du trait (en pixels)
        couleur (str): le nom d'une couleur (en anglais)
        remplissage (str) : nom de la couleur de remplissage.
    Sortie :
        La fonction ne renvoie rien, elle dessine un caré à la position (x, y)
        les paramètres epaisseur et  couleur spécifient l'épaisseur et␣
 ↪lacouleur du trait.
        Le paramètre remplissage spécifie la couleur de replissage.
        Si remplissage est une chaine vide alors on ne remplira pas
    """



##########################
#  programme principal   #
# appels de la fonction  #
##########################
carre(-100, -100, 100,3,"blue","lightblue")
carre(100, -100, 100,3,"red","")

# affichage du dessin
turtle.done()
```



Vous devez avoir obtenu ceci :

Exercice: reproduire ce dessin :



ATTENTION : vous devez utiliser votre fonction carre, et donc vous n'avec aucune fonction du module turtle à utiliser ici.

seulement VOTRE fonction.

- La largeur de chaque carré est 100
- Pour les couleurs (bordure/remplissage): > blue/lightblue, > red/salmon > dark-green/lightseagreen !!!

[ ]:
```
# votre code ici... (seulement 3 lignes, plus une pour appeler turtle.done())
```

1.0.2 D'autres fonctions: Rectangles, Triangle et disques

Fonction rectangle: adaptez votre fonction carre

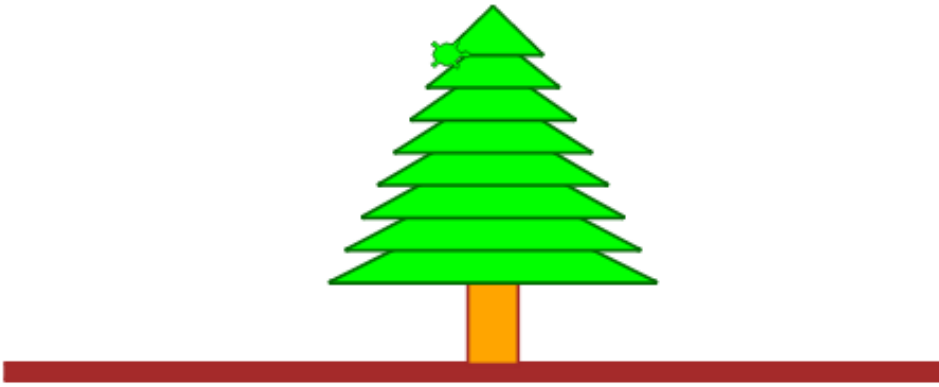
Votre fonction carre prend en paramètres:

- x et y : coordonnées du coin inférieur droit
- cote : longueur des coté
- epaisseur: epaisseur du trait
- couleur : couleur du trait

- couleur_remplissagge : couleur de remplissage

Pour dessiner un rectangle il faudra un peu adapter la liste des arguments, et modifier le code.

Une fois votre fonction écrite, testez la !



Essayez de reproduire ce dessin :

Dimensions du rectangle à gauche :30x100 dimensions du rectangleà droite : 40x70

A vous de jouer !

[ ]:
```
# recopiez la fonction carre puis modifiez la de façon a obtenir une fonction␣
 ↪rectangle
def rectangle(...) :

##########################
#  programme principal   #
# appels de la fonction  #
##########################
# a compléter également pour obtenir le dessin demandé
```

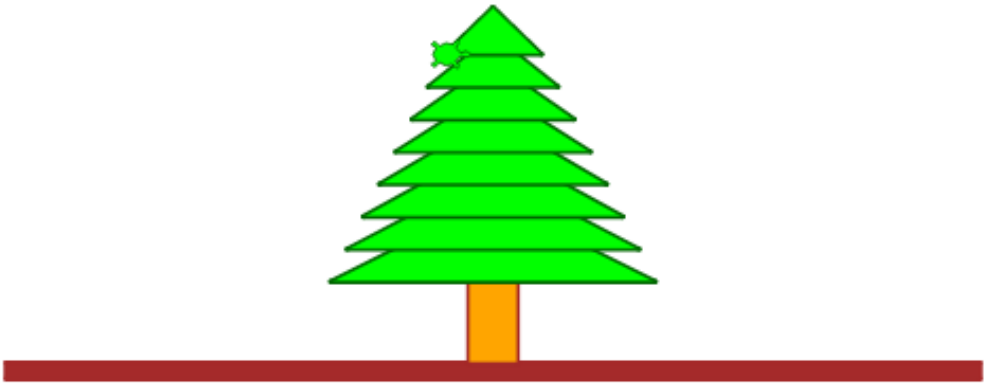
Vérifiez que vous avez bien obtenu le dessin demandé avec seulement deux appels de rectangle plus un appel de done.

Fonction triangle: adaptez la fonction carre

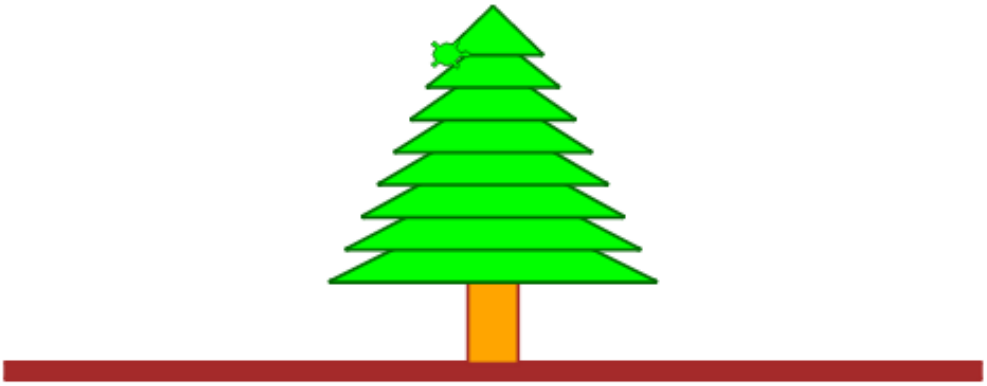
Nous allons créer une fonction qui dessine des triangles isocèles suivant ce schéma:

Le point de départ, en bas à gauche, a pourcoordonnées (x, y).

La fonction turtle.goto() permet de déplacer la tortue vers B dont les coordonnées peuvent être calculées, puis vers C et retour en A.



En conservant les paramètres utlisés pour les rectangles, (mais en remplaçant rectangle par triangle) vous obtiendrez ceci :



A vous de jouer !

[ ]:
```
# Ecrire votre fonction triangle ici



############################
#  programme principal   #
# appels de la fonction  #
############################
```

Vérifiez que vous avez bien obtenu le dessin demandé avec seulement deux appels de triangle plus un appel de done.

1.0.3 Quelques petites astuces:

accélérer le code : Voir les mouvements de la tortue aide pour réaliser les exercices, mais cela ralenti fortement nos codes.

Vous pouvez ajouter au tout début du code, pour que le dessin soit beaucoup plus rapide :

```
turtle.animation("off")
```

Cacher la tortue A la fin du dessin, la tortue reste visible. Il est possible de la cacher :

```
turtle.hideturtle()
```

Triangle pointe en bas Si vous indiquez une hauteur négative pour le triangle, le dessin sera identique sauf que la pointe du triangle sera vers le bas.

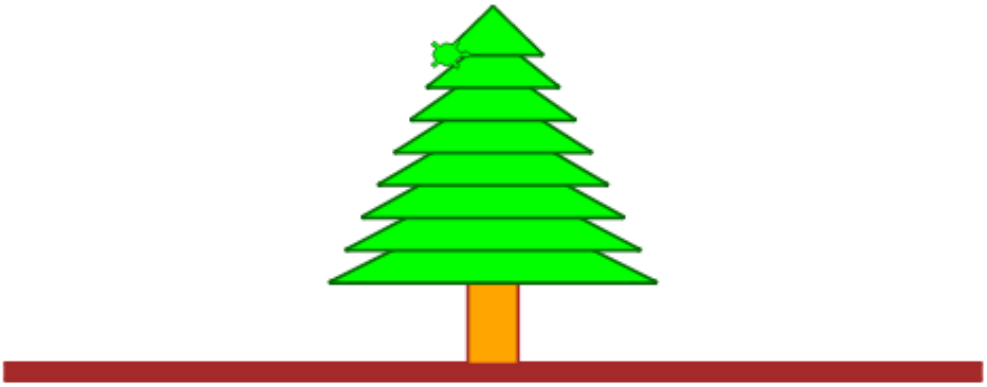
Superpositions Chaque objet dessiné recouvre les précédents.

Il est ainsi possible de créer de nouvelles formes.

code exemple :

[ ]:
```
turtle.animation("off")
turtle.hideturtle()


fond = "silver"
# on dessine un fond uni :
rectangle(-250,-100,500,300,3,"black",fond)

# on dessine un triangle
triangle(-50,0,100,-50,3,"black","palegreen")
# puis un second, identique, 20 pixels plus bas et de la couleur du fond
triangle(-50,20,100,-50,3,"silver","silver")
# ce second triangle va masquer le premier !

# on recommence un peu plus bas, mais avec des hauteurs positives :
triangle(-50,-40,100,50,3,"black","palegreen")
triangle(-50,-60,100,50,3,"silver","silver")

turtle.done()
```

Dessinez-moi un arbre! En utilisant vos fonctions, reproduire ce dessin:



Le tronc :

- largeur 30 / hauteur 70 couleurs : brown/orange

Les triangles :

- le premier fait une largeur 90 et une hauteur 47
- a chaque triangle, on réduit la hauteur de 10 et lalargeur de 3

Positions Tous les objets sont centrés horizontalement sur l'abcisse x = 0

A vous de trouver comment, à chaque objet, déterminer la valeur de l'abcisse.

Pour les ordonnées : + Le tronc débute à y = 0 + Le premier triangle à y = 50 + a chaque triangle y augmente de 20

1.1 contrainte :

`Vous devez faire les triangles dans une boucle, avec un seul appel de la fonction triangle dans`

A vous de jouer !

```
[ ]: # votre code ici
```

Auteur Jean-Louis Thirot