1) Lancer l'éditeur **Mu.**

2) Enregistrer votre travail dans un fichier ayant pour nom **tp1** et pour emplacement votre **dossier personnel**.

3) Taper les lignes suivantes :

```
from turtle import *
write(position( ))
forward (100)
write(position( ))
```

4) Cliquer sur le bouton **Lancer** pour **afficher** le résultat de ce code.

5) En déduire le rôle de chaque commande : **write**, **position** et **forward**.

Commande	Rôle
write	…… ……
position	…… ……
forward	…… ……

6) Remplacer la commande **forward** par **backward**, que remarquez-vous ?

……

7) En déduire le rôle de la commande **backward**.

……

8) Ajouter les lignes suivantes au début du programme et **observer** les résultats.

```
title ("mon premier code")
bgcolor ("blue")
color ("white")
speed ("slowest")
width (5)
```

9) En déduire le rôle de chaque commande : **title** , **bgcolor**, **color** , **speed** et **width**.

Commande	Rôle
title	…… ……
bgcolor	…… ……
color	…… ……
speed	…… ……
width	…… ……

10) Ajouter les lignes suivantes à la fin du programme et **observer** les résultats.

left (90) **forward** (100)

11) En déduire le rôle de la commande **left**.

……

12) Remplacer la commande **left** par **right**, que remarquez-vous ?

……

13) En déduire le rôle de la commande **right**.

……

14) Effectuer les modifications nécessaires qui permettent de dessiner un carré.

…… …… …… …… …… …… …… …… ……

Correction

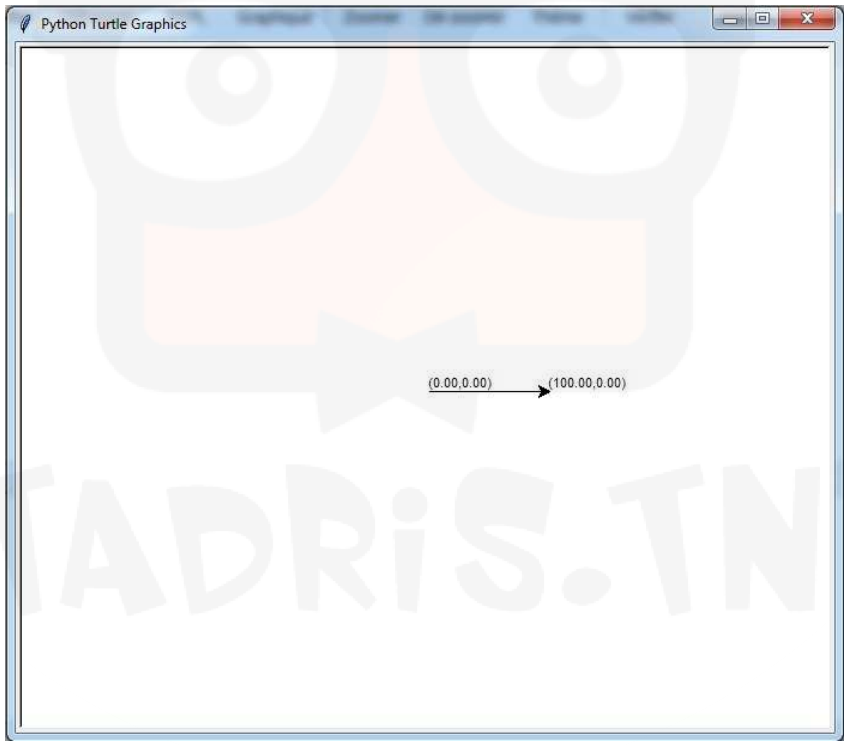
1) Lancer l'éditeur **Mu.**

2) Enregistrer votre travail dans un fichier ayant pour nom **tp1** et pour emplacement votre **dossier personnel**.

3) Taper les lignes suivantes :

```
from turtle import *
write(position( ))
forward (100)
write(position( ))
```

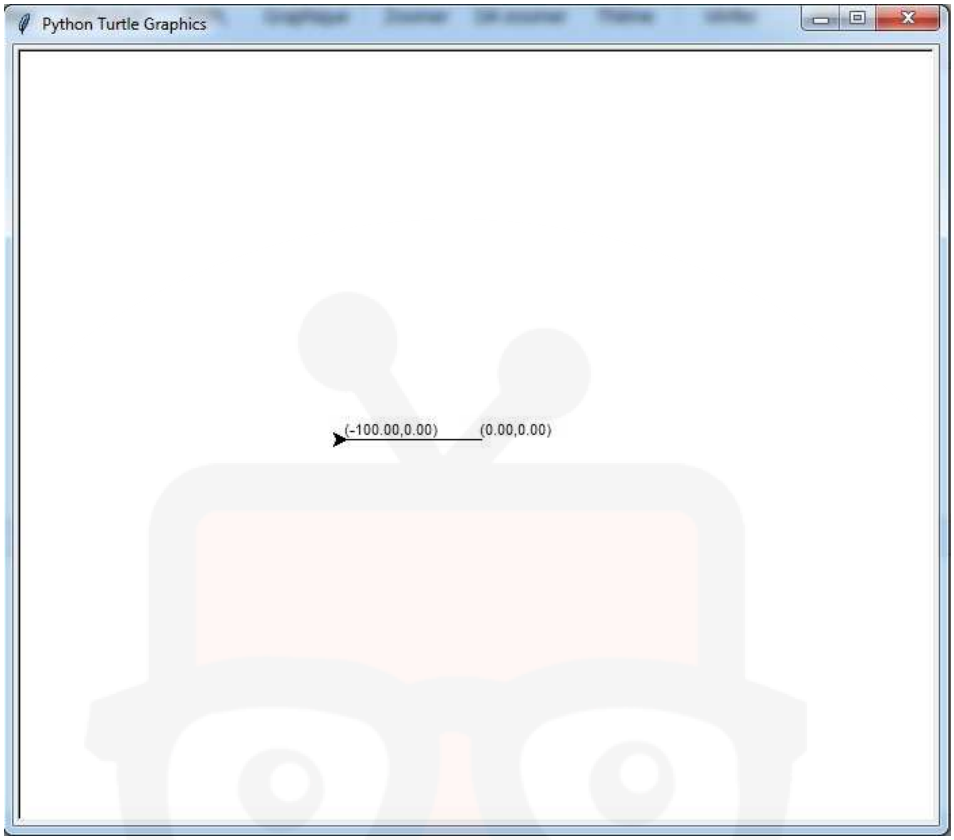
4) Cliquer sur le bouton **Lancer** pour **afficher** le résultat de ce code.



5) En déduire le rôle de chaque commande : **write**, **position** et **forward**.

Commande	Rôle
write	**write (texte) : affiche un texte donné.**
position	**position ( ) : renvoie les coordonnées actuelles du crayon.**
forward	**forward (distance) : avance d'une distance donnée.**

6) Remplacer la commande **forward** par **backward**, que remarquez-vous ?

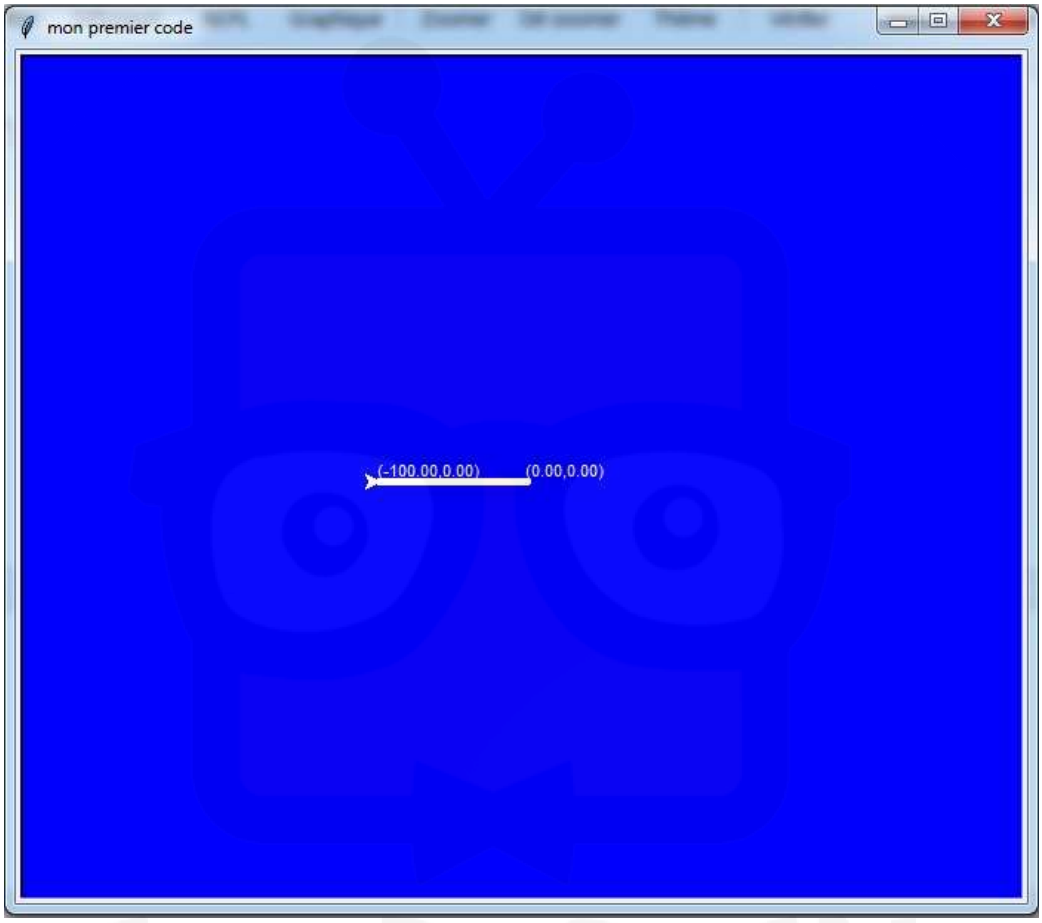


7) En déduire le rôle de la commande **backward**.

backward (distance) : recule d'une distance donnée.

8) Ajouter les lignes suivantes au début du programme et **observer** les résultats.

```
title ("mon premier code")
bgcolor ("blue")
color ("white")
speed ("slowest")
width (5)
```



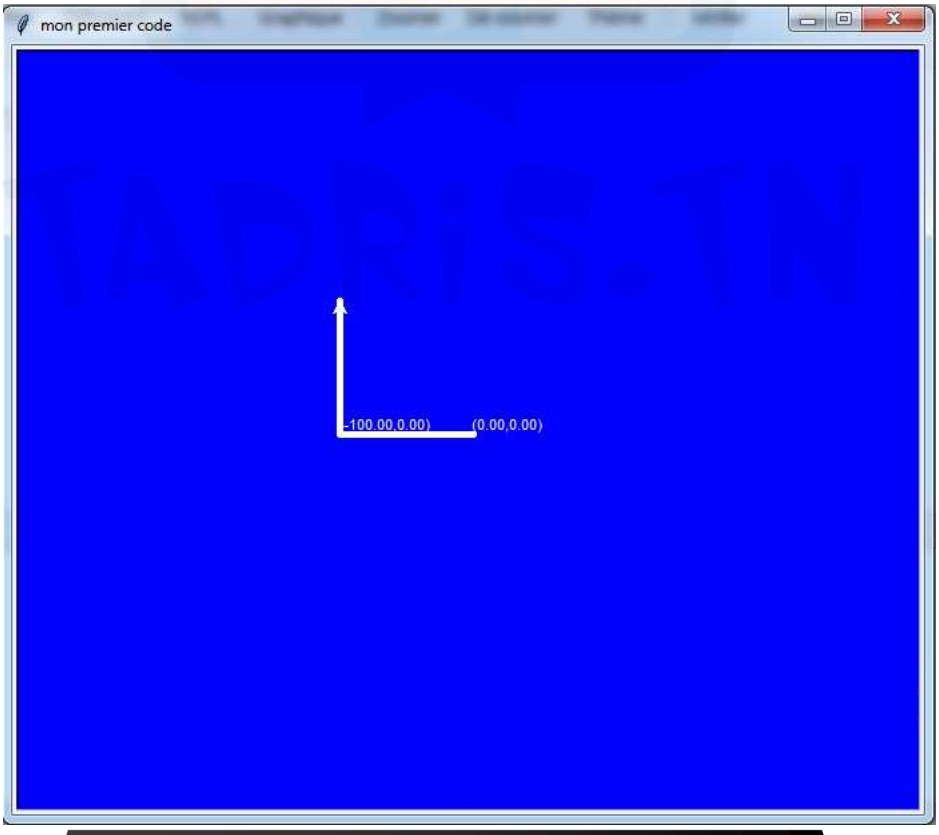
9) En déduire le rôle de chaque commande : **title**, **bgcolor**, **color**, **speed** et **width**.

Commande	Rôle
title	**<u>title (titre)</u> :** **donne un titre à la fenêtre (par défaut le titre est Turtle Graphics).**
bgcolor	**<u>bgcolor (couleur)</u> :** **détermine la couleur du l'arrière-plan de fenêtre (noir par défaut).** **<u>couleurs proposées</u> :** **"blue","red","green","yellow" ,"brown","black","white","pink", "orange","purple","grey"**

color	**<u>color (couleur)</u> :** **détermine la couleur du tracé (noir par défaut)** **<u>couleurs proposées</u> :** **"blue","red","green","yellow" ,"brown","black","white","pink", "orange","purple","grey"**
speed	**<u>speed (vitesse)</u> :** **choisit la vitesse à laquelle se déplace le curseur.** **<u>vitesses proposées</u> :** o **"slowest" => le plus lent** o **"slow" => lent** o **"normal" => normal** o **"fast" => rapide** o **"fastest" => le plus rapide**
width	**<u>width (épaisseur)</u> : choisit l'épaisseur du tracé**

10) Ajouter les lignes suivantes à la fin du programme et **observer** les résultats.

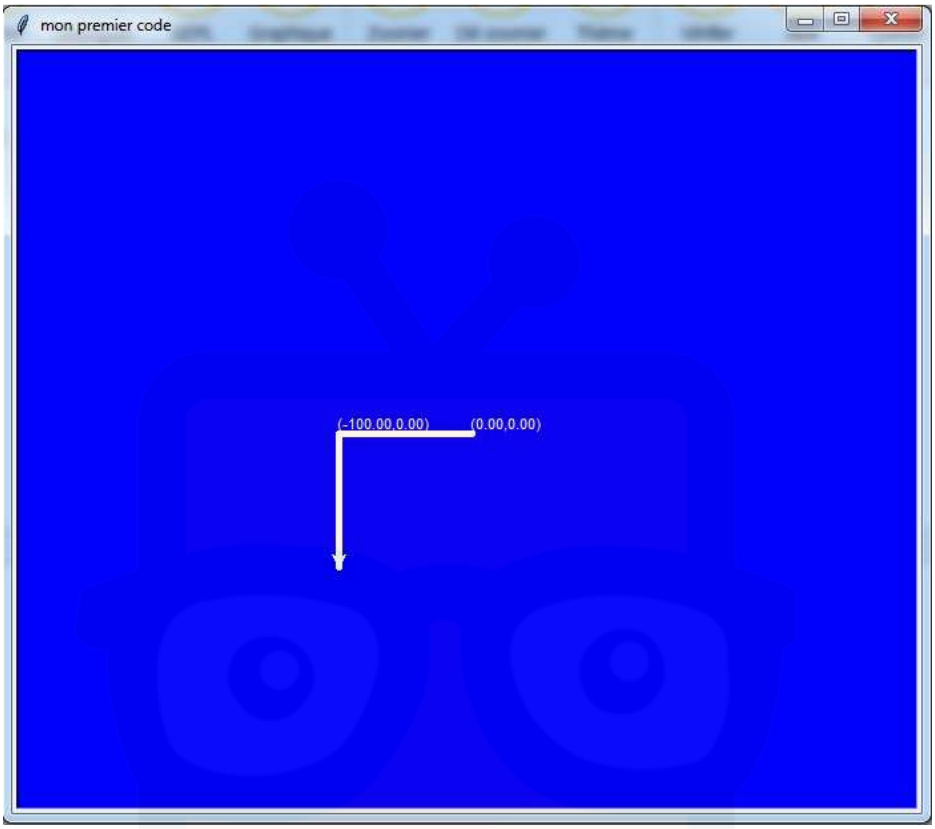
```
left (90)
forward (100)
```



11) En déduire le rôle de la commande **left**.

left (angle) : tourne à gauche d'un angle donné (en degrés).

12) Remplacer la commande **left** par **right**, que remarquez-vous ?



13) En déduire le rôle de la commande **right**.

right (angle) : tourne à droite d'un angle donné (en degrés).

14) Effectuer les modifications nécessaires qui permettent de dessiner un carré.

```
from turtle import *
title ("mon premier code")
bgcolor ("blue")
color ("white")
speed ("slowest")
width (5)
write (position( ))
backward (100)
write (position( ))
right (90)
forward (100)

# à ajouter
left (90)
forward (100)
left (90)
forward (100)
```

Travaux pratiques n°2

1) Lancer l'éditeur **Mu.**

2) Enregistrer votre travail dans un fichier ayant pour nom **tp2** et pour emplacement votre **dossier personnel**.

3) Taper les lignes suivantes :

```
from turtle import *
shape ("turtle")
width (5)
pencolor ("red")
circle (100)
```

4) Cliquer sur le bouton **Lancer** pour **afficher** le résultat de ce code.

5) Remplacer la commande **circle** (100) par **circle** (100,90), que remarquez-vous ?

……

6) En déduire le rôle de chaque commande : **shape**, **pencolor** et **circle**.

Commande	Rôle
shape	…… ……
pencolor	…… ……
circle	…… ……

7) Ajouter les lignes suivantes à la fin du programme et **observer** les résultats.

```
goto (-150,0)
circle (100)
```

8) En déduire le rôle de la commande **goto**.

……

9) Ajouter la ligne suivante **avant** la commande **goto** et **observer** les résultats.

```
up ( )
```

10) En déduire le rôle de la commande **up**.

……

11) Ajouter la ligne suivante **<u>avant</u>** la commande **goto** et **observer** les résultats.

down ( )

12) En déduire le rôle de la commande **down**.

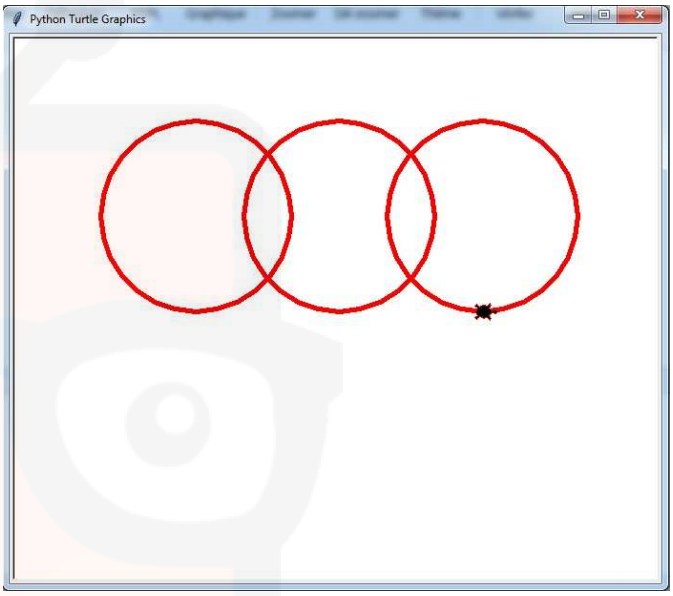
……

13) Proposer une solution pour améliorer l'affichage.

……

14) Effectuer les modifications nécessaires qui permettent de dessiner la forme suivante.

……
……
……
……
……
……
……
……
……
……
……
……



Correction

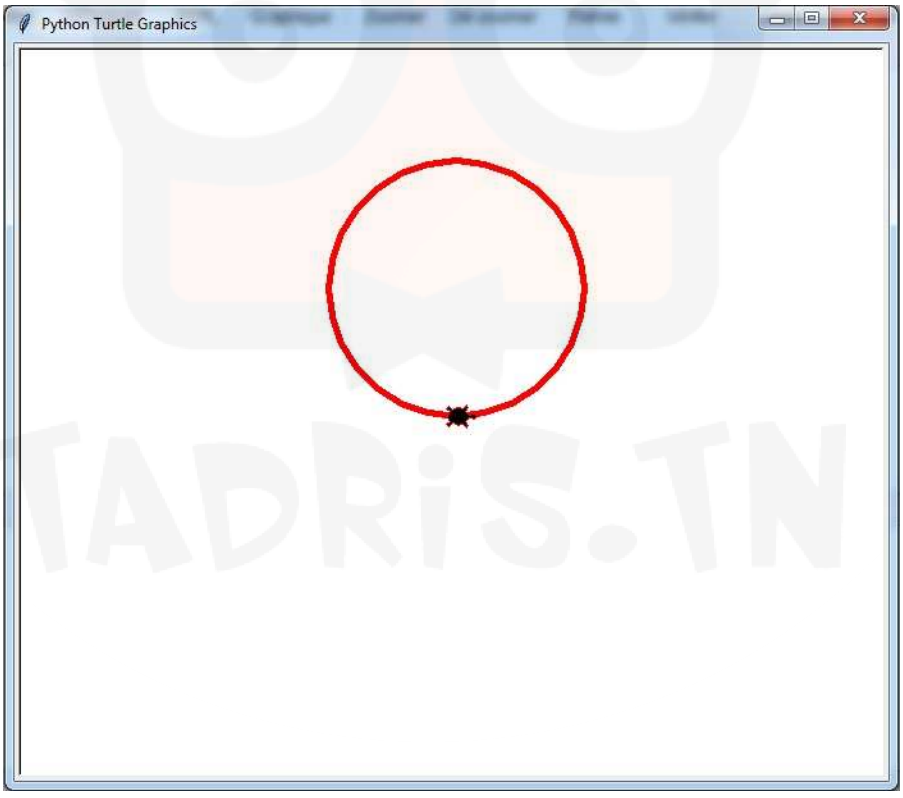
1) Lancer l'éditeur **Mu.**

2) Enregistrer votre travail dans un fichier ayant pour nom **tp2** et pour emplacement votre **dossier personnel**.

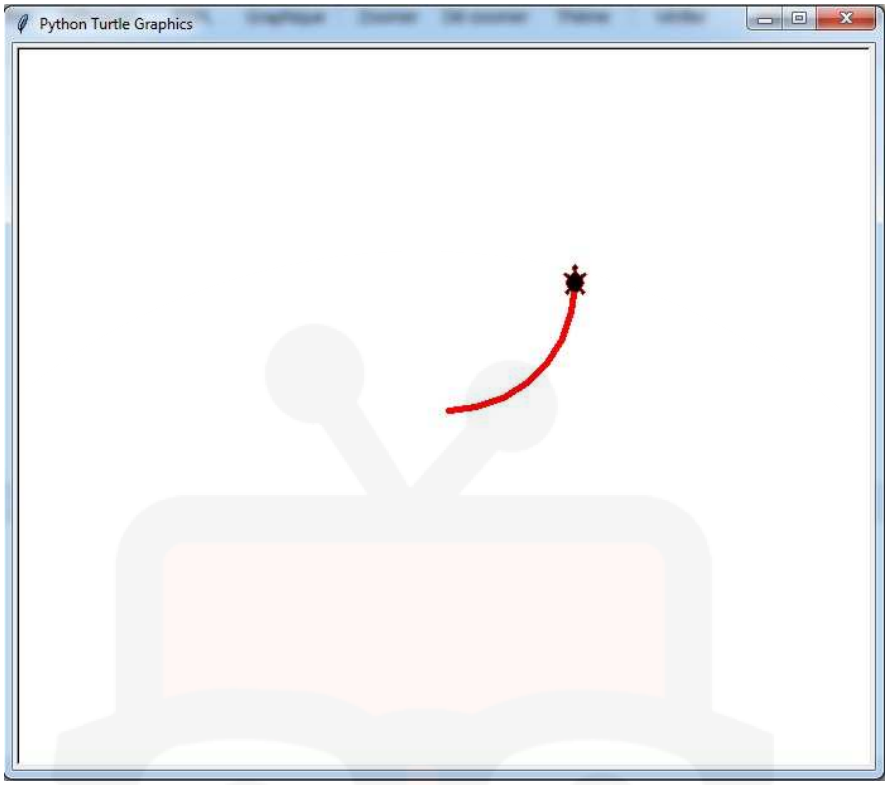
3) Taper les lignes suivantes :

```
from turtle import *
shape ("turtle")
width (5)
pencolor ("red")
circle (100)
```

4) Cliquer sur le bouton **Lancer** pour **afficher** le résultat de ce code.



5) Remplacer la commande **circle** (100) par **circle** (100,90), que remarquez-vous ?

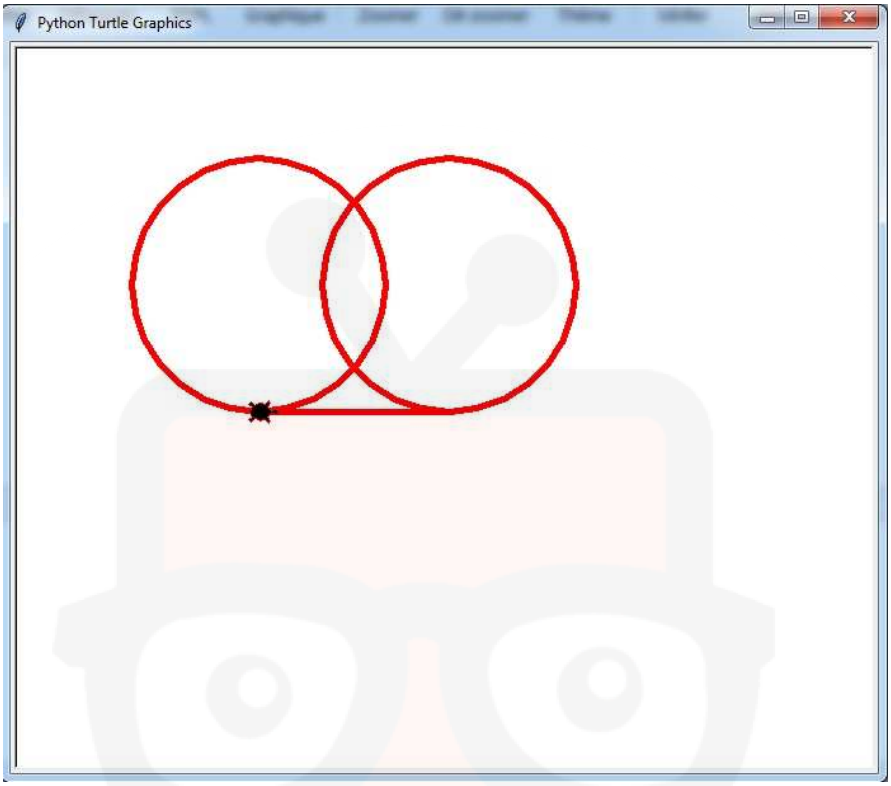


6) En déduire le rôle de chaque commande : **shape**, **pencolor** et **circle**.

Commande	Rôle
shape	**shape (forme) : forme possible : "classic" ou "turtle"**
pencolor	**pencolor (couleur) : couleur du tracé**
circle	**circle (x,[y]) : trace un cercle de rayon x, dans la continuité du tracé précédent. Possibilité d'arc de cercle avec une valeur d'angle y (sans crochets pour l'utiliser).**

7) Ajouter les lignes suivantes à la fin du programme et **observer** les résultats.

```
goto (-150,0)
circle (100)
```

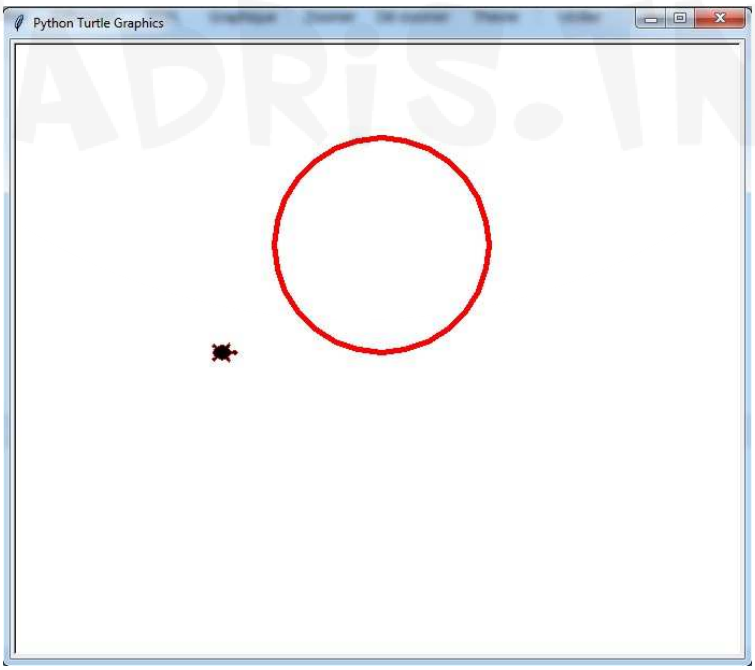


8) En déduire le rôle de la commande **goto**.

goto (x,y) : déplace le crayon jusqu'aux coordonnées (x,y).

9) Ajouter la ligne suivante **avant** la commande **goto** et **observer** les résultats.

```
up ( )
```

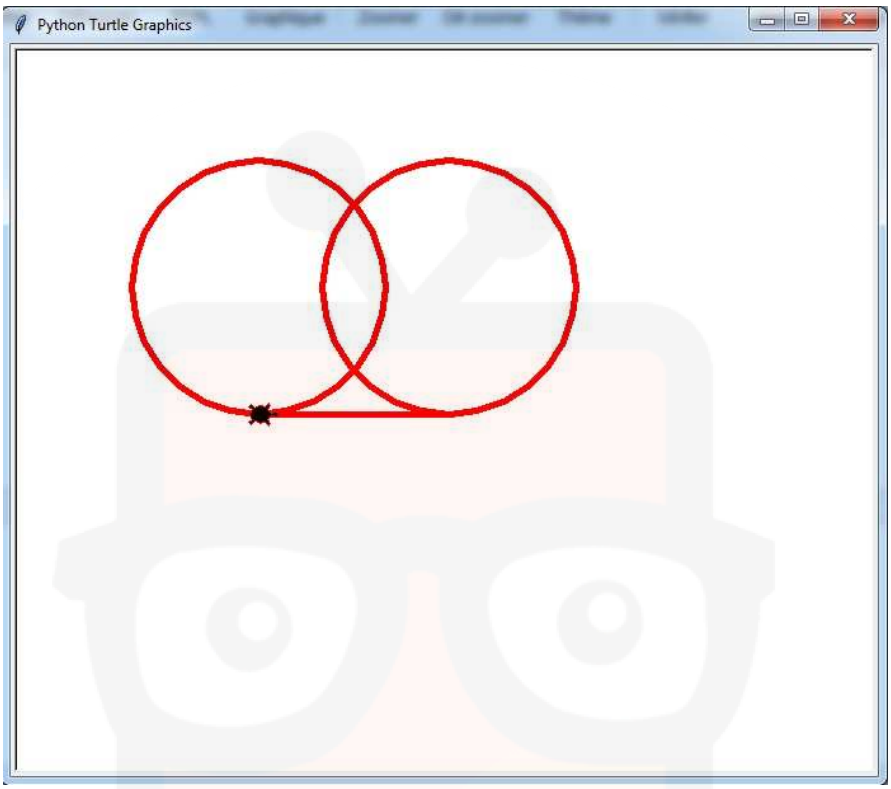


10) En déduire le rôle de la commande **up**.

up ( ) : relève le crayon (pour le déplacer sans dessiner).

11) Ajouter la ligne suivante **avant** la commande **goto** et **observer** les résultats.

```
down ( )
```

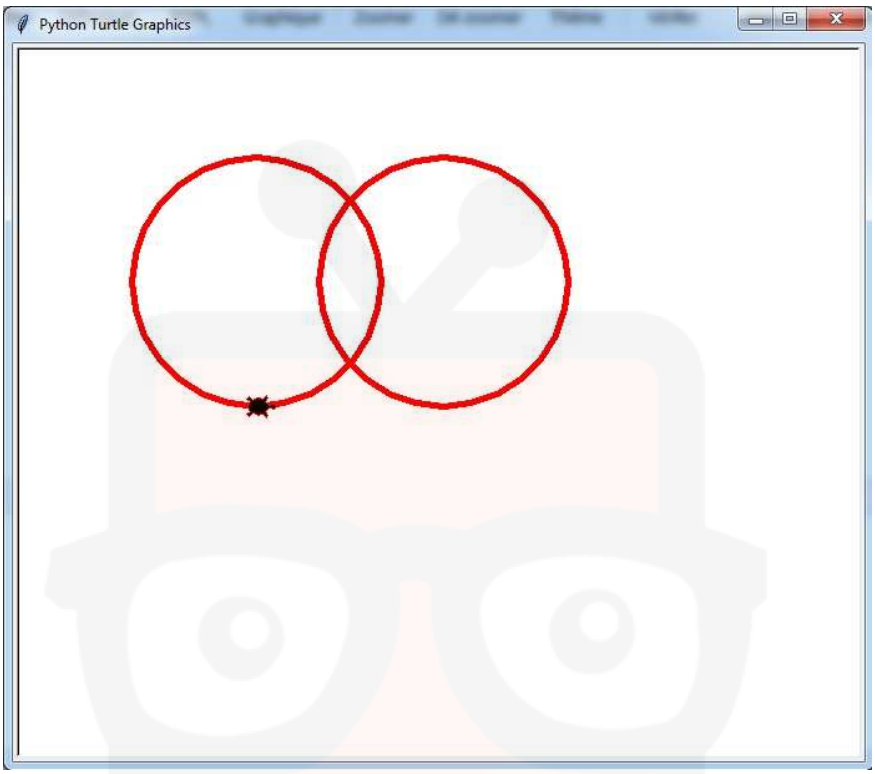


12) En déduire le rôle de la commande **down**.

down ( ) : abaisse le crayon (pour recommencer à dessiner).

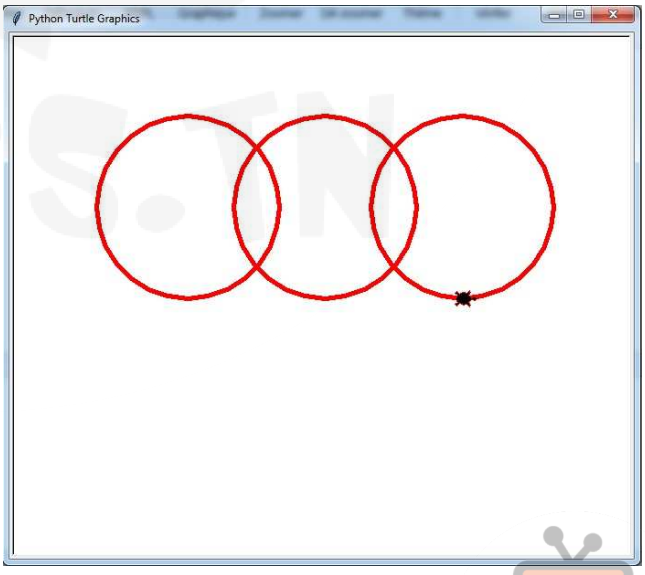
13) Proposer une solution pour améliorer l'affichage.

```
up ( )
goto (-150,0)
down ( )
```



14) Effectuer les modifications nécessaires qui permettent de dessiner la forme suivante.

```
from turtle import *
shape ("turtle")
width (5)
pencolor ("red")
circle (100)
up ( )
goto (-150,0)
down ( )
circle (100)
# à ajouter
up ( )
goto (150,0)
down ( )
circle (100)
exitonclick ( )
```



Travaux pratiques n°3

1) Lancer l'éditeur **Mu.**

2) Enregistrer votre travail dans un fichier ayant pour nom **tp3** et pour emplacement votre **dossier personnel**.

3) Taper les lignes suivantes :

```
from turtle import *
speed (5)
shape ("turtle")
color ("red", "black")
pensize (4)
```

4) Cliquer sur le bouton **Lancer** pour **afficher** le résultat de ce code.

5) Compléter le script précédent pour qu'il' trace une marche d'escalier comme ci-dessous, à partir du point de coordonnées **(-150,-150)**, avec une hauteur et une largeur **50** pixels.

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

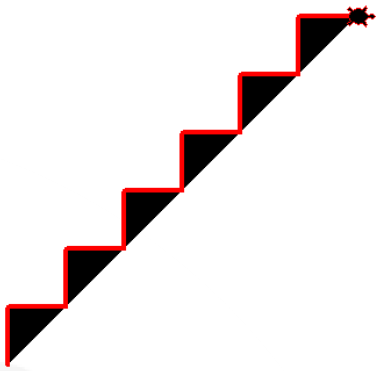
6) Mettre les lignes produisant la forme entre **begin_fill** ( ) et **end_fill** ( ) et **observer** les résultats.

7) En déduire le rôle de chaque commande : **begin_fill** et **end_fill**.

Commande	Rôle
begin_fill	....
end_fill	....

8) Compléter le script précédent pour qu'il' trace un escalier comme ci-dessous, à partir de la position courante, avec une hauteur et une largeur **50** pixels.

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
```



9) Que remarquez-vous ?

.....

.....

10) Ajouter la ligne suivante **avant** les lignes qui produisent la forme d'une marche et **observer** les résultats.

```
for i in range (3) :
```

11) En déduire le rôle de la structure **for**.

.....

12) Qu'est ce qui il faut changer pour obtenir notre escalier ?

.....

13) En déduire le rôle de la commande **range**.

.....

14) Effectuer les modifications nécessaires qui permettent de tracer un triangle équilatéral de côté 100 pixels et dont un sommet a pour coordonnées (0,0).

```
.....
.....
.....
.....
.....
.....
.....
.....
```

Correction

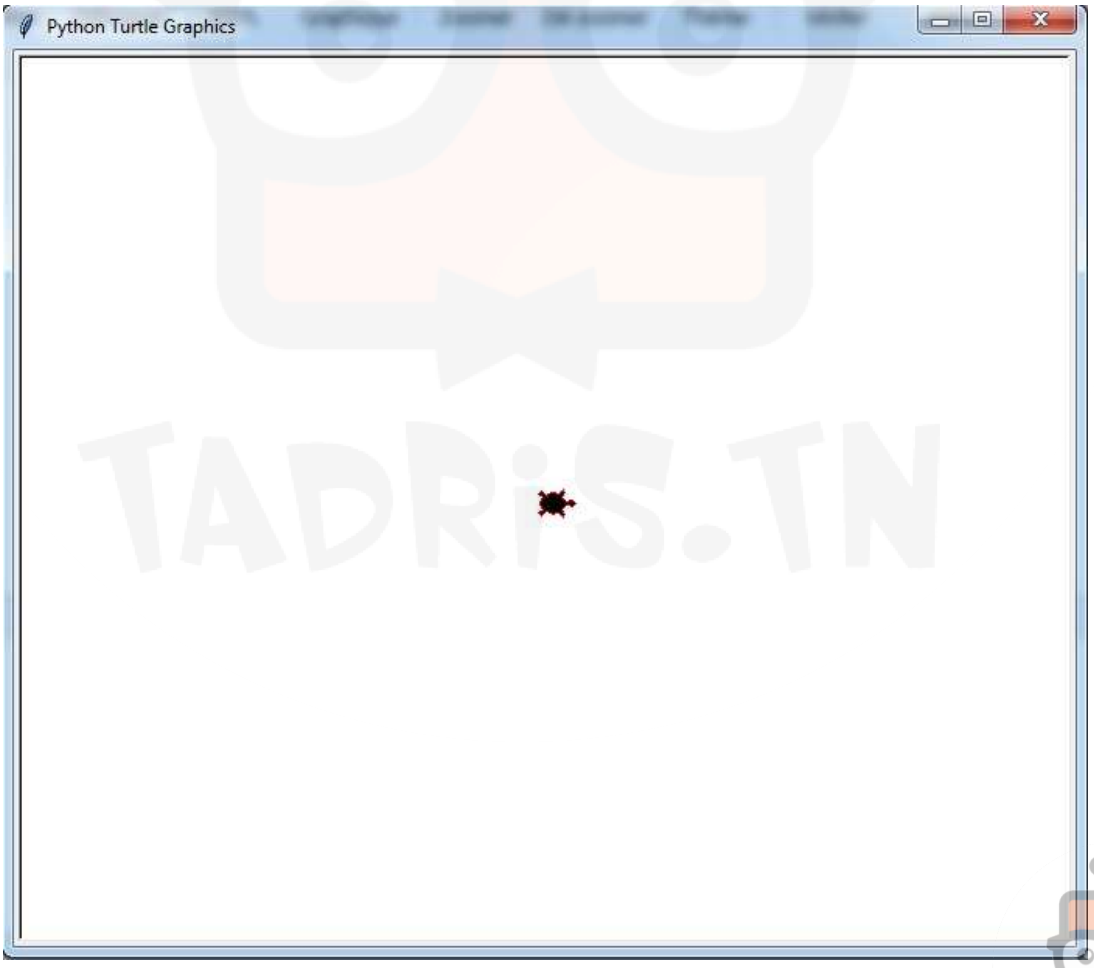
1) Lancer l'éditeur **Mu.**

2) Enregistrer votre travail dans un fichier ayant pour nom **tp3** et pour emplacement votre **dossier personnel**.

3) Taper les lignes suivantes :

```
from turtle import *
speed (5)
shape ("turtle")
color ("red", "black")
pensize (4)
```

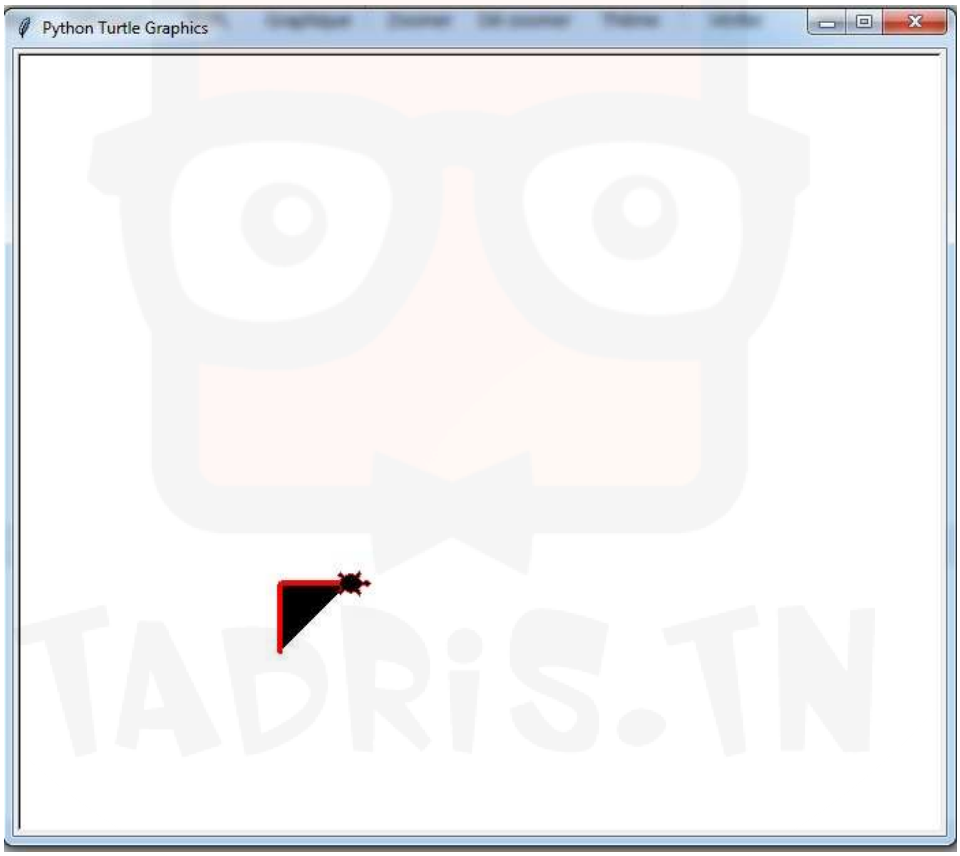
4) Cliquer sur le bouton **Lancer** pour **afficher** le résultat de ce code.



5) Compléter le script précédent pour qu'il' trace une marche d'escalier comme ci-dessous, à partir du point de coordonnées **(-150,-150)**, avec une hauteur et une largeur **50** pixels.

```
up ( )
goto (-150,-150)
down ( )
left (90)
forward (50)
right (90)
forward (50)
```

6) Mettre les lignes produisant la forme entre **begin_fill** ( ) et **end_fill** ( ) et **observer** les résultats.

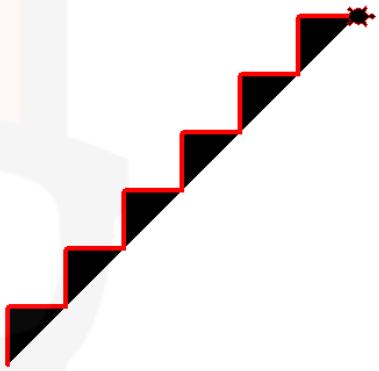


7) En déduire le rôle de chaque commande : **begin_fill** et **end_fill**.

Commande	Rôle
begin_fill	**remplit un contour fermé à l'aide de la couleur sélectionnée.**
end_fill	**on peut aussi utiliser fill (1) et fill (0).**

8) Compléter le script précédent pour qu'il' trace un escalier comme ci-dessous, à partir de la position courante, avec une hauteur et une largeur **50** pixels.

```
begin_fill ( )
left (90)
forward (50)
right (90)
forward (50)

left (90)
forward (50)
right (90)
forward (50)

left (90)
forward (50)
right (90)
forward (50)

left (90)
forward (50)
right (90)
forward (50)

left (90)
forward (50)
right (90)
forward (50)

left (90)
forward (50)
right (90)
forward (50)
end_fill ( )
```



9) Que remarquez-vous ?

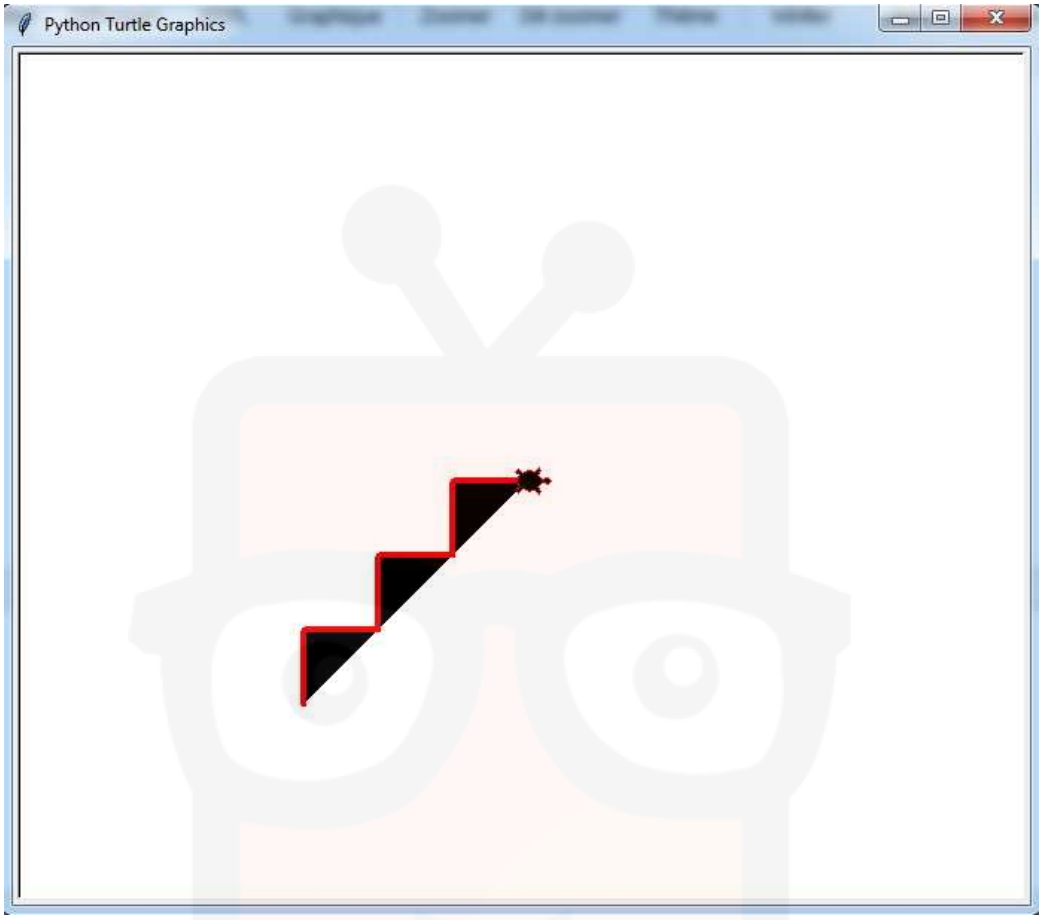
Script très long.

Le bloc suivant se répète 6 fois :

left (90)
forward (50)
right (90)
forward (50)

10) Ajouter la ligne suivante **avant** les lignes qui produisent la forme d'une marche et **observer** les résultats.

```
for i in range (3) :
```



11) En déduire le rôle de la structure **for**.

La boucle for permet d'exécuter une itération un nombre de fois connu.

12) Qu'est ce qui il faut changer pour obtenir notre escalier ?

range (6)

13) En déduire le rôle de la commande **range**.

La fonction **range** permet de générer une liste d'entiers.
L'appel de fonction **range** (n) renvoie la liste des entiers de 0 inclus à n exclu.

Exemples :

range (5) produit la liste [0,1,2,3,4]
range (0) renvoie une liste vide [ ]

Autres paramétrages possibles:

range (n_debut_inclus, n_fin_exclu)
range (n_debut_inclus, n_fin_exclu, increment_de_saut)

Exemples :

range (4,8) renvoie [4, 5, 6, 7]
range (4,10,2) renvoie [4, 6, 8]

14) Effectuer les modifications nécessaires qui permettent de tracer un triangle équilatéral de côté 100 pixels et dont un sommet a pour coordonnées (0,0).

```
from turtle import *
speed (5)
shape ("turtle")
color ("red", "black")
pensize (4)
up ( )
goto (0,0)
down ( )
begin_fill ( )
for i in range (3) :
   forward (100)
   left (120)
end_fill ( )
exitonclick ( )
```

Travaux pratiques n°4

1) Lancer l'éditeur **Mu.**

2) Taper le code suivant et l'**enregistrer** dans un fichier ayant pour nom **tp4** et pour emplacement votre **dossier personnel**:

```
from turtle import *
shape ("turtle")
pensize (5)
right (90)
forward (25)
```

3) Exécuter le programme précédent puis **déduire** son rôle.

……

4) Ajouter la ligne suivante **avant** la commande **right** puis **identifier** son rôle.

```
x=textinput ("direction" , "gauche/droite")
```

……

5) Taper le texte "**gauche**" dans la zone du texte qui apparut. **Que remarquez-vous ?**

……

6) Dans quel cas notre programme fonctionne correctement ?

……

7) Effectuer les modifications nécessaires pour que notre programme fonctionne correctement.

……

……

……

8) Ajouter les instructions nécessaires pour que notre programme fonctionne correctement dans tous les cas possibles.

……

……

……

……

9) Ajouter les instructions nécessaires pour que notre programme puisse faire 10 pas.

……

……

Correction

1) Lancer l'éditeur **Mu.**

2) Taper le code suivant et l'**enregistrer** dans un fichier ayant pour nom **tp4** et pour emplacement votre **dossier personnel**:

```
from turtle import *
shape ("turtle")
pensize (5)
right (90)
forward (25)
```

3) Exécuter le programme précédent puis **déduire** son rôle.

Le programme permet d'avancer un pas de 25 pixels vers la droite.

4) Ajouter la ligne suivante **avant** la commande **right** puis **identifier** son rôle.

```
x=textinput ("direction" , "gauche/droite")
```

textinput (titre, message) : demande une chaîne à l'utilisateur dans une fenêtre contextuelle.
titre : le titre de la fenêtre.
message : le texte affiché dans la fenêtre.

5) Taper le texte "**gauche**" dans la zone du texte qui apparut. **Que remarquez-vous ?**

Lorsque qu'on tape "gauche" dans la zone du texte : le programme fait un pas vers la droite.

→ Erreur

6) Dans quel cas notre programme fonctionne correctement ?

Si le texte tapé est "droite", le programme fonctionne correctement.

7) Effectuer les modifications nécessaires pour que notre programme fonctionne correctement.

```
from turtle import *
shape("turtle")
pensize(5)
x=textinput("direction","gauche/droite")
if (x=="droite"):
    right(90)
    forward(25)
```

8) Ajouter les instructions nécessaires pour que notre programme fonctionne correctement dans tous les cas possibles.

```
from turtle import *
shape("turtle")
pensize(5)
x=textinput("direction","gauche/droite")
if (x=="droite"):
    right(90)
    forward(25)
if (x=="gauche"):
    left(90)
    forward(25)
```

```
from turtle import *
shape("turtle")
pensize(5)
x=textinput("direction","gauche/droite")
if (x=="droite"):
    right(90)
    forward(25)
elif (x=="gauche"):
    left(90)
    forward(25)
else:
    write("erreur")
```

9) Ajouter les instructions nécessaires pour que notre programme puisse faire 10 pas.

```
from turtle import *
shape("turtle")
pensize(5)
for k in range(10):
    x=textinput("direction","gauche/droite")
    if (x=="droite"):
        right(90)
        forward(25)
    elif (x=="gauche"):
        left(90)
        forward(25)
    else:
        write("erreur")
```