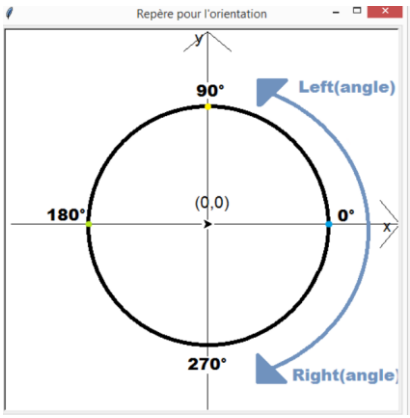
```
turtle.clear()  #Efface les dessins du crayon
turtle.reset()  #Fait de même et réinitialise le crayon turtle.forward(turtle.window_width()/3)
#Avance d'un tiers de la largeur de la fenêtre
turtle.backward(turtle.window_width()/2)  #Recule de la moitié de la largeur de la fenêtre
turtle.bk(50)  #Recule de 50px
turtle.fd(0)  #Avance de 0px, donc n'avance pas turtle.goto(100, 100)  #Position (100, 100)
turtle.setx(20)  #Position(20, 100)
turtle.sety(-80)  #Position(20, -80)
turtle.home()  #Position(0, 0) (équivalent à turtle.goto(0, 0))
 turtle.up()  #Lève le crayon
turtle.forward(150)  #Avance de 150px sans tracer
turtle.down()  #Abaisse le crayon
turtle.backward(50)  #Recule de 50px en traçant
turtle.up()  #Lève le crayon
turtle.forward(150)  #Avance de 150px sans tracer
turtle.down()  #Abaisse le crayon
turtle.backward(50)  #Recule de 50px en traçant
print(turtle.pensize())  #Affiche '1'
turtle.pensize(5.5)  #Modifie la largeur du traçage
print(turtle.pensize())  #Affiche '5.5'
turtle.setup()  #Initialise la fenêtre
print(turtle.heading())  #Affiche 0.0 : le crayon pointe vers le point bleu : Est
turtle.left(90)  #Pointe vers le point jaune : Nord
turtle.right(270)  #Pointe vers le point vert : Ouest
turtle.setheading(0)  #Pointe de nouveau vers le point bleu
turtle.setheading(-90)  #Pointe à l'opposé du point jaune : Sud
print(turtle.heading())  #Affiche '270.0'
turtle.exitonclick()  #Clique gauche pour fermer
angle = turtle.towards(0, 90)
print(angle)  #Affiche '90.0'
turtle.setheading(angle)  #Angle : 90.0
turtle.forward(90)  #Position : (0, 90); Angle : 90.0
turtle.goto(0, 90)  #Position : (0, 90); Angle : 0.0
```

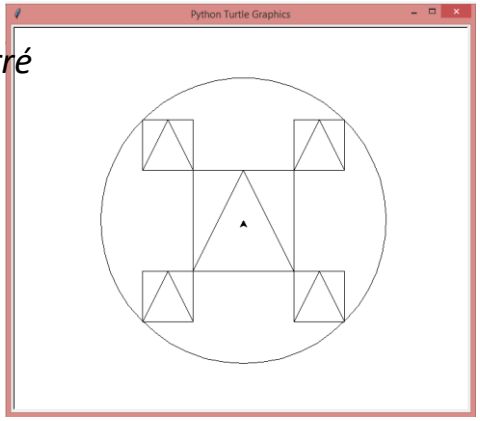



```
#Un exemple de triangle équilatéral
longueur_cote = 200
turtle.forward(longueur_cote)  #1er côté
turtle.left(360/3)  #Angle
turtle.forward(longueur_cote)  #2ème côté
turtle.left(360/3)  #Angle
turtle.forward(longueur_cote)  #3ème côté
```

```
#Un exemple de carré
longueur_cote = 200
for i in range(4):
    turtle.forward(longueur_cote)  #Côté
    turtle.left(90)  #Angle
```

```
turtle.circle(120)  #Trace un cercle de rayon 120px
turtle.circle(70, 180)  #Trace un demi-cercle de rayon 70px
turtle.circle(90, steps = 8)  #Trace un octogone de longueur 90px
turtle.circle(40, 180, 4)  #Trace la moitié d'un octogone de longueur 40px

#5 cercles du plus petit au plus grand, centre origine du repère
import turtle
    rayon, ecart = 50, 20
    for i in range(5):
        turtle.up()
        turtle.goto(0, -rayon)
        turtle.down()
        turtle.circle(rayon)
        rayon += ecart  #Augmente la valeur de rayon
    turtle.up()
    turtle.home()
    turtle.exitonclick()


import turtle

def deplacer_sans_tracer(x, y = None):
    """Fonction pour se déplacer à un point sans tracer"""
    turtle.up()
    if (isinstance(x, tuple) or isinstance(x, list)) and len(x) == 2:
        turtle.goto(x)
    else:
        turtle.goto(x, y)
    turtle.down()
def triangle_dans_carre(long_carre):

    """Fonction pour tracer un triangle à l'intérieur du carré"""
    #On prend la position du curseur qui est sur le coin bas gauche
    pos_coin_bg = turtle.position()
    #On trace les deux traits restants, la base étant déjà faite
    turtle.goto(pos_coin_bg[0]+long_carre/2, pos_coin_bg[1]+long_carre)
    turtle.goto(pos_coin_bg[0]+long_carre, pos_coin_bg[1])
```

```
def carre_avec_triangle(longueur):
    """Fonction pour tracer un carré avec un triangle à l'intérieur"""
    for i in range(4):
        turtle.forward(longueur)
        turtle.left(90)
    triangle_dans_carre(longueur)

    #On initialise les longueurs du grand carré et des petits carrés
    longueur_1, longueur_2 = 150, 75
    #On se positionne au coin bas gauche de notre futur grand carré
    deplacer_sans_tracer(-longueur_1/2, -longueur_1/2)
    #On le dessine
    carre_avec_triangle(longueur_1)
    #On prépare les valeurs des coin bas gauche des petits carrés
    coins = [(longueur_1/2, longueur_1/2),
             (-longueur_1/2-longueur_2, longueur_1/2),
             (-longueur_1/2-longueur_2, -longueur_1/2-longueur_2),
             (longueur_1/2, -longueur_1/2-longueur_2)]
    #On dessine notre quatrième petit carré
    for coin in coins:
        deplacer_sans_tracer(coin)
        carre_avec_triangle(longueur_2)
    #On retourne au centre de notre dessin
    deplacer_sans_tracer(0, 0)
    #On prend la distance entre le centre et le coin par lequel le cercle passera
    rayon = turtle.distance(longueur_1/2+longueur_2, longueur_1/2+longueur_2)
    #On se déplace et on trace notre cercle
    deplacer_sans_tracer(0, -rayon)
    turtle.circle(rayon)
    #On retourne à la maison, et on prend un angle de 90°
    deplacer_sans_tracer(0, 0)
    turtle.left(90)
    turtle.exitonclick()
turtle.dot(100, 'red')  #Imprime un point rouge d'un diamètre de 100px
turtle.dot(50, 'yellow')  #Imprime un point jaune d'un diamètre de 50px
turtle.dot(25)  #Imprime un point noir d'un diamètre de 25px
import turtle
from random import randint
COULEURS = ['black', 'grey', 'brown', 'orange', 'pink', 'purple',
            'red', 'blue', 'yellow', 'green']
if __name__ == "__main__":
    turtle.setup(650, 100)
    diametre = 15
    turtle.up(); turtle.setx(-turtle.window_width()/2+2*diametre); turtle.down()
    #Pour le nombre de couleurs disponibles
```



```
for i in range(len(COULEURS)):
    #On choisit un couleur aléatoirement
    index_choisi = randint(0, len(COULEURS)-1)
    #On imprime un point de cette couleur
    turtle.dot(diametre, COULEURS[index_choisi])
    #On supprime la couleur choisie pour éviter de la rechoisir
    del COULEURS[index_choisi]
    #On met à jour le diamètre et on se déplace pour le prochain point
    diametre += 5; turtle.up(); turtle.fd(1.5*diametre); turtle.down()
turtle.exitonclick()
```