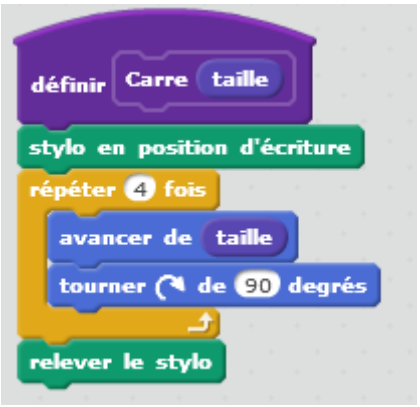
De scratch à python : 3

Exercice 1 :

Nous voulons tracer 10 carrés dont les côtés font 20 pas. L'espace entre les carrés est de 5 pas.

Créer un bloc Carre, recevant en paramètre un nombre représentant la « taille » du carré à tracer. Ce bloc s'occupe de la gestion du crayon : crayon posé ou crayon relevé.

Créer un script, qui utilise ce bloc Carre pour tracer nos 10 carrés en ligne.



Avec Python

Dans python un bloc est une fonction que l'on doit définir :

def nom_fonction(parametre1, parametre2…) :

"Texte expliquant ce que fait la fonction"

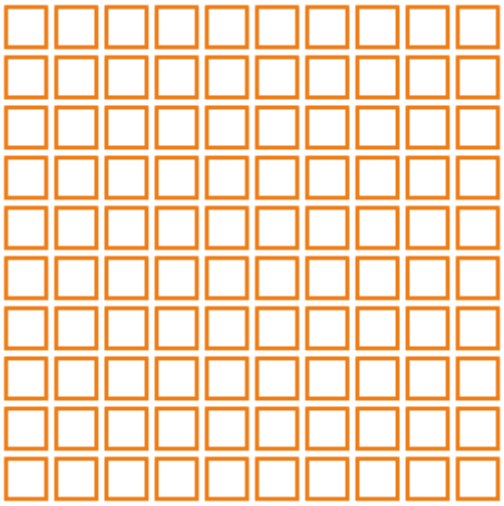
instructions de la fonction

Le programme

```
#Le bloc Carre qui reçoit en paramètre la taille du coté
def Carre(taille):
  "Dessine un carré de coté = taille"
  tu.down()           #poser le crayon
  for i in range(4):    #dessiner 4 cotés
     tu.forward(taille)
     tu.right(90)
  tu.up()         #lever le crayon
                  #A la fin du tracé, la tortue est revenue à son point de départ
                  #au coin inférieur gauche du carré
#-----------------------------------------------
import turtle as tu
#Préparation de l'environnement de la tortue
tu.speed(5)  #parametrage de la vitesse de 1 lent à 10 rapide, 0 étant la vitesse la plus rapide
tu.shape("turtle")    #choix de la forme de la tortue
#-----------------------------------------------
tu.goto(-180,0)       # aller à la position (-180,0)
tu.pencolor("red")  #choix de la couleur du crayon
tu.pensize(2)         #épaisseur du crayon
tu.clear()            #effacer tout
#---------------------------------
for j in range(10): #Dessin de 10 carrés
   Carre(20)      #appel du bloc Carre en lui indiquant la taille du coté
   tu.forward(25)  #on avance de 25 pas avant de tracer le carré suivant
tu.forward(30)     #on avance la tortue pour qu'elle ne masque pas le dernier carré
#-----------------------------------------------
tu.mainloop()
tu.bye()
```

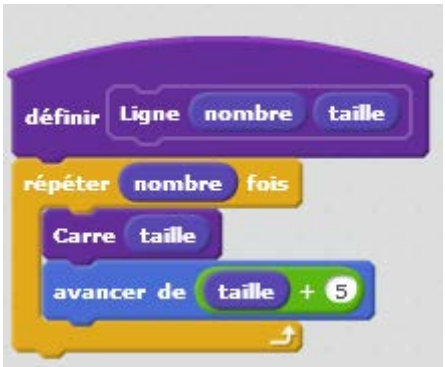
Exercice 2 :

Nous voulons maintenant créer plusieurs lignes de carrés, séparées les unes des autres de 5 pas.



- Créer un bloc Ligne, recevant en paramètre un nombre indiquant le « nombre » de carrés à tracer dans la ligne et un nombre indiquant la « taille » du carré à tracer.

- Créer un script, qui utilise ce bloc Ligne pour tracer 10 lignes séparer les unes des autres de 5 pas.



Avec Python

```
#Le bloc Carre qui reçoit en paramètre la taille du coté
def Carre(taille):
  "Dessine un carré de coté = taille"
  tu.down()           #poser le crayon
  for i in range(4):
     tu.forward(taille)
     tu.right(90)
  tu.up()             #lever le crayon
#-------------------------------------------------------------
#Le bloc Ligne qui reçoit en paramètre le nombre de carrés à tracer et la taille du coté
def Ligne(nombre, taille):
   "Dessine une ligne de 'nombre' carrés de coté = taille"
   for i in range(nombre):    #on trace nombre carrés
      Carre(taille)        #chaque carré a un coté = taille
      tu.forward(taille + 5) #A la fin du tracé d'un carré, la tortue est revenue
                    #au coin inférieur gauche du carré. Il faut donc avancer
                    #la tortue d'un nombre de pas égal à taille + 5, si l'on veut
                    #que les carrés soient séparés de 5 pas
#-------------------------------------------------
import turtle as tu
#Préparation de l'environnement de la tortue
tu.speed(0)          #parametrage de la vitesse 0 : vitesse la plus rapide
tu.shape("turtle")    #choix de la forme de la tortue
#-----------------------------------------------
tu.goto(-180,180)     # aller à la position (-180,180)
tu.pencolor("green")   #choix de la couleur du crayon
tu.pensize(2)         #épaisseur du crayon
tu.clear()           #effacer tout
```

```
#---------------------------------
for j in range(10):        #on trace 10 lignes de carrés
    Ligne(10,20)           #tracé d'une ligne de 10 carrés de coté = 20
    #Avant de tracer la ligne suivante il faut descendre la tortue et
    #la remettre à gauche de l'écran.
    #25 = taille du coté d'un carré + 5 pas de séparation entre les lignes
    tu.sety(tu.ycor()- 25)
    tu.setx(-180)
#---------------------------------------------
tu.mainloop()
tu.bye()
```

Exercice 3 :

1) Créer un bloc Hexagone recevant en paramètre la taille d'un coté.

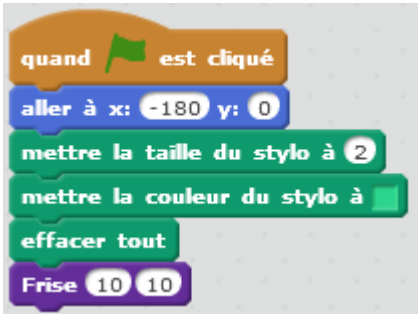
   Tracer la figure suivante à l'aide de ce bloc

Avec Python

#Le bloc **Hexagone** qui reçoit en paramètre la taille du coté

```
def Hexagone(taille):
  "Dessine un hexagone de coté = taille"
  tu.down()            #poser le crayon
  for i in range(6):    #dessiner 6 cotés
     tu.forward(taille)
     tu.left(60)
  tu.up()                 #lever le crayon
                          #A la fin du tracé, la tortue est revenue à son point de départ
                           #au coin inférieur gauche de l'hexagone
#-----------------------------------------------
import turtle as tu
#Préparation de l'environnement de la tortue
tu.speed(5)
tu.shape("turtle")    #choix de la forme de la tortue
#-----------------------------------------------
tu.goto(-180,0)      # aller à la position (-180,0)
tu.pencolor("red")  #choix de la couleur du crayon
tu.pensize(2)          #épaisseur du crayon
tu.clear()               #effacer tout
#---------------------------------
for j in range(10):    #Dessin de 10 hexagones
   Hexagone(10)      #appel du bloc Hexagone en lui indiquant la taille du coté
   tu.forward(30)     #on avance de 30 pas avant de tracer l'hexagone suivant suivant
#-----------------------------------------------
tu.mainloop()
tu.bye()
```

2) Créer un bloc Frise utilisant le bloc Hexagone et recevant en paramètre le nombre d'hexagones à créer et la taille d'un coté de ces héxagone.

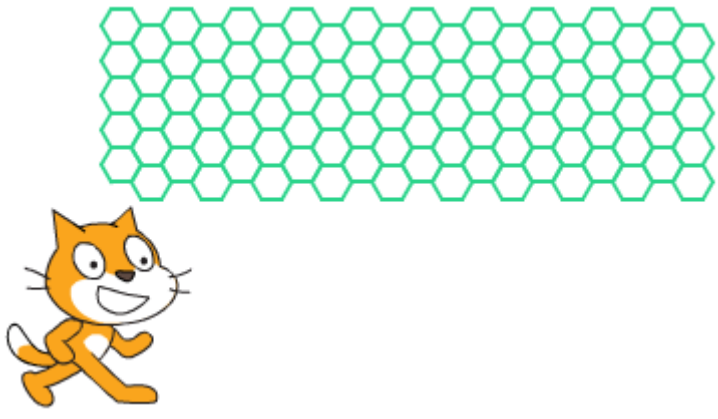


Avec Python

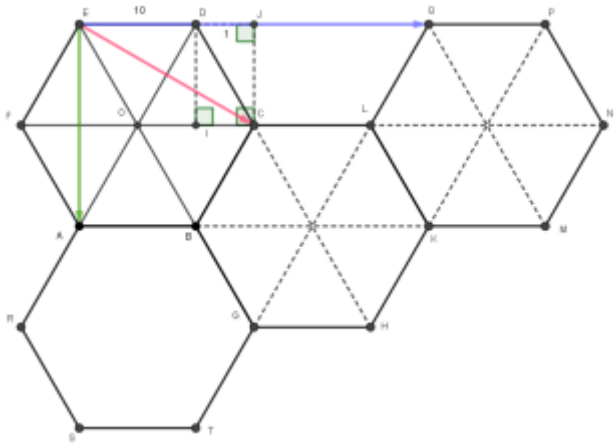
```
#Le bloc Hexagone qui reçoit en paramètre la taille du coté

def Hexagone(taille):
  "Dessine un hexagone de coté = taille"
  tu.down()            #poser le crayon
  for i in range(6):   #dessiner 6 cotés
     tu.forward(taille)
     tu.left(60)
  tu.up()              #lever le crayon
                       #A la fin du tracé, la tortue est revenue à son point de départ
                       #au coin inférieur gauche de l'hexagone
#-----------------------------------------------------------
#Le bloc Frise qui reçoit en paramètre le nombre d'hexagones à tracer et la taille du coté

def Frise(nombre, taille):
   "Dessine une ligne du futur pavage: nombre=nombre d'hexagones à dessiner, taille=coté de l'hexagone"
   tu.down()
```

```
    for i in range(nombre):     #dessiner nombre hexagones
        Hexagone(taille)
        tu.forward(taille*3)
    tu.up()
#-----------------------------------------------
import turtle as tu
#Préparation de l'environnement de la tortue
tu.speed(5)
tu.shape("turtle")
#-----------------------------------------------
tu.goto(-180,0)
tu.pencolor("red")
tu.pensize(2)
tu.clear()
#--------------------------------
Frise(10,10)
#-----------------------------------------------
tu.mainloop()
```

3) Créer un bloc pavage qui réalise le pavage suivant : il reçoit en paramètre le nombre de lignes du pavage, le nombre de colonnes, la taille d'un hexagone, les coordonnées x et y où commence le pavage.



La difficulté ici est de positionner en hauteur (coordonnée y) chacune des lignes.



Sur le schéma ci-dessus, la hauteur DI est telle que

$$CD^2 = DI^2 + CI^2$$

$$I \text{ est le milieu du segment } OC \text{ qui est égale à } taille$$

$$DI^2 = CD^2 - CI^2$$

$$DI^2 = taille^2 - \left(\frac{taille}{2}\right)^2$$

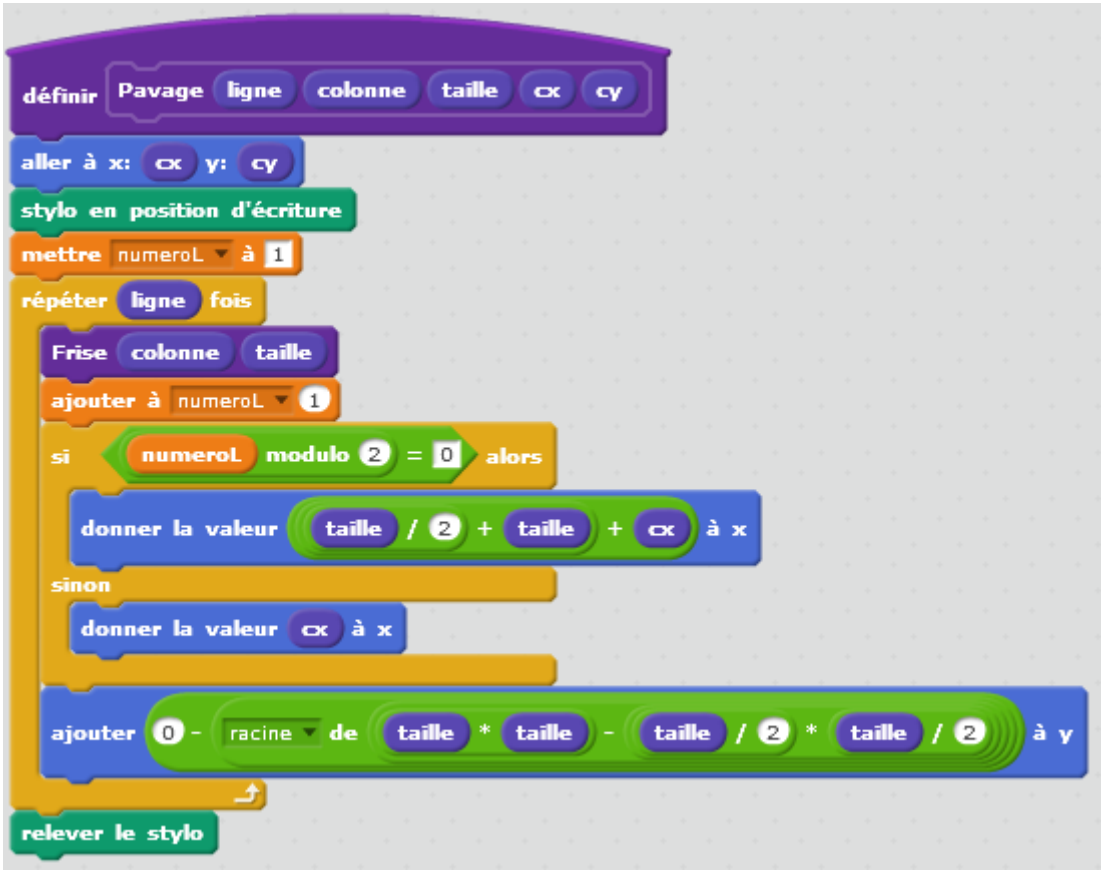
$$DI = \sqrt{taille^2 - \left(\frac{taille}{2}\right)^2}$$

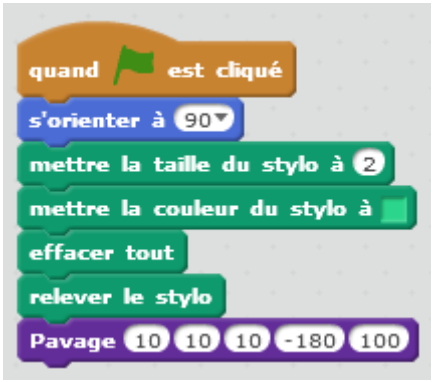
A chaque nouvelle ligne, il faut donc diminuer la valeur actuelle de y, d'une valeur égale à DI

Pour la position en x de la nouvelle ligne, il faut savoir si on a à faire à une ligne dont le numéro est pair ou impair. Si la ligne est impaire, la ligne commence à la valeur de départ de x.

Si la ligne est paire, la nouvelle ligne commence à :

$$valeur\ départ\ x + taille + \frac{taille}{2}$$





Avec Python

Nous avons besoin ici de deux fonctions du module math :

- La fonction sqrt(val) : renvoie la racine carrée de val
- La fonction pow(x, y) : renvoie x à la puissance y, soit x^y

Pour savoir si le numéro d'une ligne est pair ou impair nous utiliserons comme avec Scratch une variable numeroL que nous incrémentons de 1, à chaque fois que l'on a fini de tracer une ligne.

L'opération modulo (reste de la division entière) s'écrit en python % : 4 % 2 = 0 et 9 % 2 = 1

En python le test d'une condition du type 

s'écrit :

if **condition** :

Instructions à exécuter si la condition est vraie

suite du programme

En python le test d'une condition du type 

s'écrit :

if **condition** :

Instructions à exécuter si la condition est vraie

else :

Instructions à exécuter si la condition est fausse

suite du programme

Tester si un nombre est égale à une valeur s'écrit : if nombre **==** valeur :

Le programme

```
#Le bloc Hexagone qui reçoit en paramètre la taille du coté
def Hexagone(taille):
  "Dessine un carré de coté = taille"
  tu.down()
  for i in range(6):
     tu.forward(taille)
     tu.left(60)
  tu.up()
```

```
#Le bloc Frise qui reçoit en paramètre le nombre d'héxagone à tracer et la taille du coté
def Frise(nombre, taille):
    "Dessine une ligne du futur pavage: nombre=nombre d'hexagones à dessiner, taille=coté de l'héxagone"
    tu.down()
    for i in range(nombre):     #dessiner nombre hexagone
        Hexagone(taille)
        tu.forward(taille*3)
    tu.up()

#Le bloc Pavage qui reçoit en paramètre le nombre de lignes et de colonnes à tracer
#le taille d'un coté du motif hexagone, la position de départ du pavage
def Pavage(ligne, colonne,taille, cx, cy):
    "Dessine un pavage dont le motif est hexagonal"
    tu.goto(cx,cy)
    tu.down()
    numeroL = 1   #variable nous indiquant le numéro de la ligne à tracer
    for j in range(ligne):     #dessiner nombre hexagone
        Frise(colonne,taille)
        numeroL=numeroL+1 #numéro de la prochaine ligne à tracer
        #calculer la nouvelle position en x de la prochaine ligne à tracer
        #----tester si le numéro de la ligne suivante à tracer est paire ou impaire
        if numeroL % 2 == 0:
            tu.setx(cx+taille+taille/2)
        else:
            tu.setx(cx)
        #calculer la nouvelle position en y de la prochaine ligne à tracer
        tu.sety(tu.ycor()-sqrt((pow(taille,2)-pow(taille//2,2))))
    tu.up()
```

```
#-----------------------------------------------
import turtle as tu
from math import sqrt, pow
#Préparation de l'environnement de la tortue
tu.speed(5)
tu.shape("turtle")
#-----------------------------------------------
tu.setheading(0)
tu.pencolor("red")
tu.pensize(2)
tu.clear()
tu.up()
#-------------------------------
Pavage(10,10,10,-180,100)
#-----------------------------------------------
tu.mainloop()
tu.bye()
```