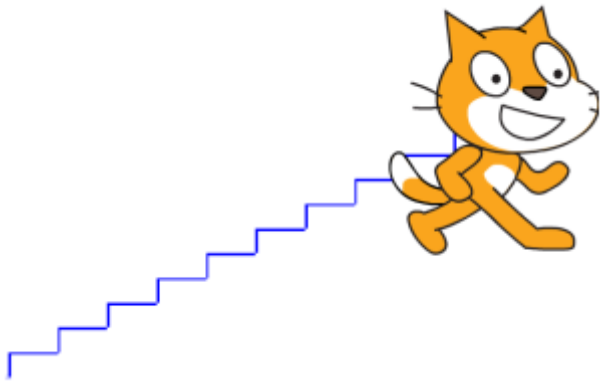
De scratch à python : 2

Exercice 1 :

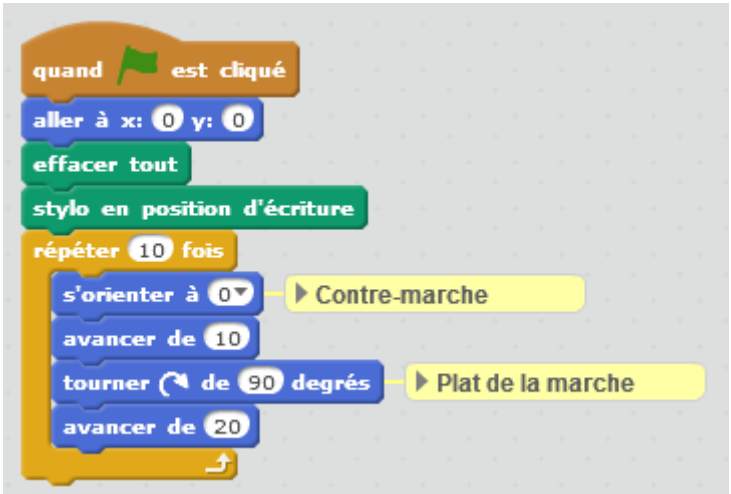
Trace un escalier, comme sur cette figure. À chaque marche, Scratch monte de 10 puis avance de 20.



Blocs utiles.

- Le bloc le plus utile sera le bloc répéter 10 fois. Toutes les instructions placées à l'intérieur de ce bloc seront répétées 10 fois.

- Autres blocs déjà vus : s'orienter à 0° (vers le haut), s'orienter à 90° (vers la droite) ... Et aussi aller à $x = 0$, $y = 0$, effacer tout, stylo en position d'écriture.



Avec Python

Le bloc 

se traduit par une boucle for :

for i in range(10):

La variable i va prendre succécivement les valeurs de 0 à 9. A chaque nouvelle valeur de i, python exécute les instructions composant la boucle. Ces instructions sont celles qui sont indentées sous l'instruction for.

for i in range(10) :

    Instruction 1

    Instruction 2

    …

    Instruction n

Suite du programme

Le programme :

```
import turtle as tu
tu.mode('logo')
tu.goto(0,0)     # aller à la position (0,0)
tu.clear()       #effacer tout
tu.down()        #poser le crayon
#---------------------------------
for i in range(10):
   tu.setheading(0) #s'orienter vers le haut
   tu.forward(10)   #avancer de 10
   tu.right(90)     #tourner à droite de 90°
   tu.forward(20)   #avancer de 20
#---------------------------------
tu.mainloop()
tu.bye()
```

Complément :

La fonction range est utilisée pour générer une séquence de nombres.

Syntaxe : range([Début,] Fin [,Pas])

Les valeurs entre crochet ne sont pas obligatoires.

Début : point de départ de la séquence. Si cette valeur n'est pa indiquée, le point de départ sera 0.

Fin : obligatoire, indique la fin de la séquence. Cette fin ne fera pas partie de la séquence.

Pas : pas de parcours de la séquence. Si cette valeur n'est pa indiquée, le pas est de 1 ou -1.

Exemple

range(5) génère les nombres 0, 1, 2, 3, 4

range(2,8) génère les nombres 2, 3, 4, 5, 6, 7

range(-2,2) génère les nombres -2, -1, 0, 1

range(2,8,2) génère les nombres 2, 4, 6

range(8,2, -2) génère les nombres 8, 6, 4

range(-100, -95) génère les nombres -100, -99, -98, -97, -96

Exercice 2 :

Trace un octogone (polygone à 8 cotés). Change de couleur à chaque côté.

Blocs utiles.

• Tourner à gauche de 45 degrés
• Ajouter 10 à la couleur du stylo

Nous allons répéter 8 fois la même séquances d'instructions :

- Tracer un coté
- Tourner vers la gauche de 45 degrés
- Changer la couleur du crayon

Au bout des 8 fois, le lutin aura fait un tour complet (8 fois 45 = 360 °) et se retrouvera à sa position de départ.

Avec Python

Pour définir une couleur nous avons le choix entre plusieurs méthodes : donner le nom de la couleur, donner sa valeur en héxadécimale, donner son code RVB (rouge, vert, bleu).

Le nom des couleurs (un petit extrait):

"black", "white", "grey", "red", "orange", "green", "blue"

"yellow", "brown", "cyan", "violet"

Le code hexadécimal d'une couleur : pour le rouge par exemple cette valeur est #FF0000
Pour le jaune : #FFFF00, pour le violet : #EE82EE etc.

Le code RVB : les couleurs sont définies par trois valeurs allant de 0 à 255, qui représentent les composantes Rouge, Vert et Bleu.
Le rouge est codé par (255,0,0), le jaune par (255,255,0), le violet par (238,130,238) etc.

tu.pencolor("red") #mettre la couleur du stylo à rouge → 

En utilisant la valeur héxadécimale : **tu.pencolor("#FF0000")**

En utilisant les valeurs RVB : (255,0,0)

tu.colormode(255) #indique la valeur maximale des composantes R,V,B de la couleur

tu.pencolor(255,0,0)

Pour changer la couleur de chaque coté du polygone, on peut par exemple, pour chacun des tracés, générer un nombre entier aléatoire compris entre 0 et 255, bornes comprises, pour chaque composante RVB.

Pour avoir accès aux fonctions générant des nombres aléatoires, nous devons importer le module **random**. Dans ce module nous ne nous servirons que de la fonction **randint** qui génère des nombres aléatoires entiers.

Le programme :

```
import turtle as tu
from random import randint
tu.colormode(255) #indique la valeur maximale des composantes R,V,B de la couleur
tu.pensize(3)          #Epaisseur du trait
#--------------------------------
#choix d'une couleur aléatoire
r = randint(0,255)
v = randint(0,255)
b = randint(0,255)
#--------------------------------
tu.goto(0,0)       # aller à la position (0,0)
tu.clear()           #effacer tout
tu.setheading(0)  #orientation à droite
tu.pencolor(r,v,b)  #choix de la couleur du crayon
tu.down()             #poser le crayon
#--------------------------------
for i in range(8): #on trace 8 cotés
   tu.forward(50)
   tu.left(45)
   #--------------------------------
   #choix d'une nouvelle couleur aléatoire
   #les composantes R,V,B ne doivent pas dépasser 255
   r = randint(0,255)
```

```
    v = randint(0,255)
    b = randint(0,255)
    tu.pencolor(r,v,b)
#---------------------------------
tu.mainloop()
tu.bye()
```

Complément :

Syntaxe : randint(Début, Fin)

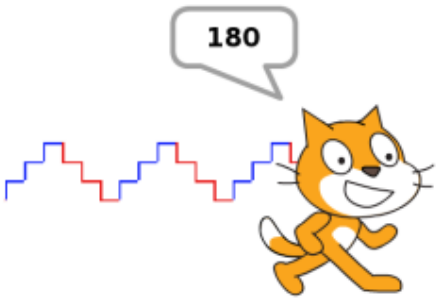
Début et Fin doivent être des nombres entiers.

La fonction renvoie une valeur entière N comprise entre Début et Fin, bornes comprises.

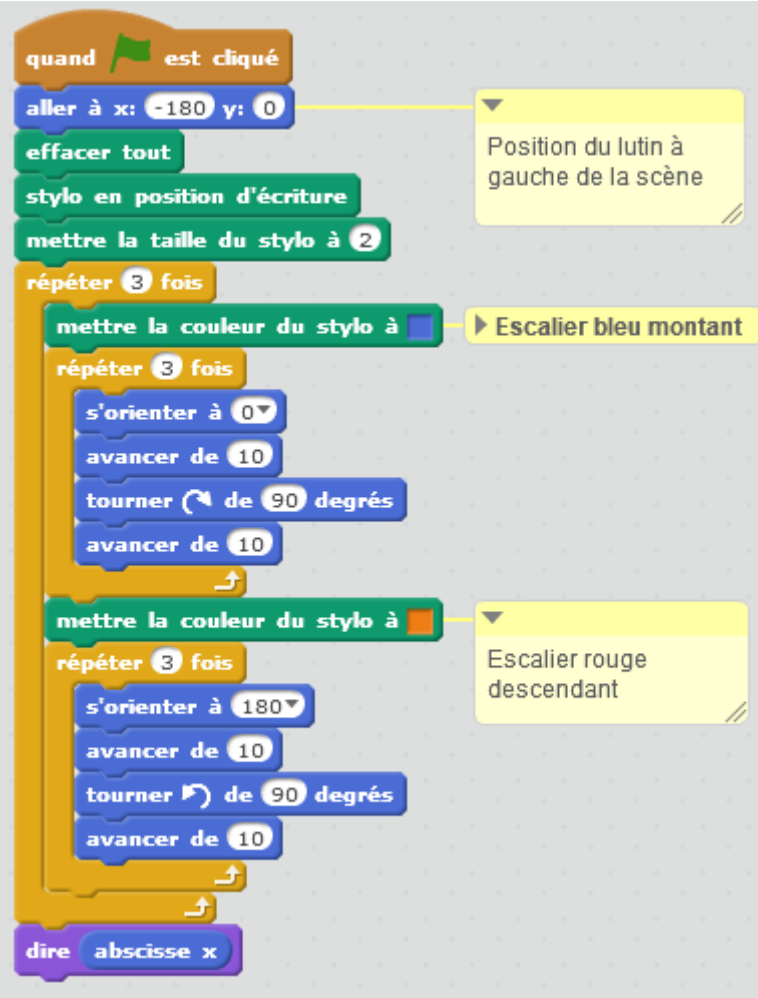
Début ≤ N ≤ Fin

Exercice 3 :

Trace des escaliers comme sur la figure.



- On répète trois fois : le chat monte de 10 puis avance de 10 (escalier bleu).
- On répète trois fois : le chat descend de 10 puis avance de 10 (escalier rouge).
- On répète ces deux opérations trois fois.
- De plus, tu peux changer la couleur du trait et afficher la valeur de l'abscisse x de Scratch lorsqu'il s'arrête.



Avec Python

Nous n'utiliserons pas dans cet exercice le mode logo, car on nous demande d'afficher la coordonnée x de la tortue en fin de parcours. En mode logo, la valeur des coordonnées de la tortue ne sont pas calculables.

Il faut donc se souvenir, que dans le mode standard, l'orientation vers le haut est 90 et non plus 0, et vers le bas 270 ou -90 au lieu de 180.

```
import turtle as tu
tu.goto(-180,0)     # aller à la position (-180,0)
tu.clear()          #effacer tout
tu.down()           #poser le crayon
tu.pensize(2)       #épaisseur du crayon
```

```
#---------------------------------
for j in range(3):
    tu.pencolor("blue")  #choix de la couleur du crayon
    #---------------------------------
    for i in range(3):
        tu.setheading(90) #s'orienter vers le haut
        tu.forward(10)    #avancer de 10
        tu.right(90)      #tourner à droite de 90°
        tu.forward(10)    #avancer de 10

    tu.pencolor("red")  #choix de la couleur du crayon
    #---------------------------------
    for i in range(3):
        tu.setheading(270) #s'orienter vers le bas
        tu.forward(10)     #avancer de 10
        tu.left(90)        #tourner à droite de 90°
        tu.forward(10)     #avancer de 10
#---------------------------------
#Ecriture des coordonnées de la tortue
cor= tu.xcor()      #coordonnée x de la tortue
tu.up()             #on lève le crayon pour pouvoir écrire au dessus du dessin
tu.hideturtle()     #on cache la tortue
tu.goto(10,100)     #on déplace la tortue au point où l'on veut écrire
tu.write(cor,move=False, align="left", font=("Arial", 18, "normal"))
#---------------------------------
tu.mainloop()
tu.bye()
```

Complément : écriture dans la fenêtre de la tortue

Syntaxe :

`turtle.write(arg, move=False, align="left", font=("Arial", 8, "normal"))`

arg – texte à écrire sur l'écran de la tortue

move – True/False

align – une des valeurs "left", "center" or right"

font – un triplet (nom de la police, taille de la police, "normal" ou "italic" ou "bold")

Cette fonnction écrit le texte **arg**, à l'emplacement de la tortue. Ce texte sera aligné suivant ce qui est indiqué dans le paramètre **align**. La police utilisée, sa taille, son type dépend du paramètre **font**. Si **move** est à True, le crayon est déplacé vers le coin en bas à droite du texte. La valeur par défaut est False.