ATELIER : De Scratch à Python

Les instructions d'affectation et l'utilisation de fonctions

Exemple de situation 1 : un programme de calcul.

Script Scratch	Pseudo-code
	Saisir x a ← 4 x + 3 b ← 2 x + 5 y ← a – b Afficher y
Python sans fonction	Python avec fonction

Scénario activité élèves :

Les élèves sont sur ordinateur avec une IDE Python. Deux codes Python leur seraient donnés, l'un avec la liste des instructions sans fonction et l'autre avec une fonction. On leur fait constater que la version sans fonction est quasiment identique au pseudo-code donné. On leur fait aussi constater que les deux codes produisent le même résultat, on montre aussi l'utilisation du mode console pour la version avec fonction et que cette utilisation permet de s'affranchir des problèmes d'entrées-sorties. On institutionnalise l'écriture d'une fonction en Python.

Travail pour les stagiaires :

Compléter le tableau ci-dessus en donnant un code Python avec puis sans fonction, tester les programmes en cas de doute. Utiliser le mode console pour tester et appeler la fonction.

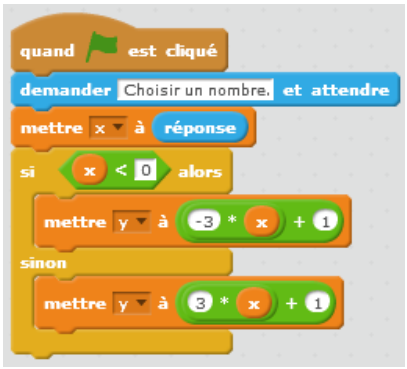
Les instructions conditionnelles

Exemple de situation 2 : une fonction affine définie par morceaux.

Exemple donné :

<table>
<tr>
<td>Fonction mathématique

f est la fonction définie sur $\mathbb{R}$ par :
$\begin{cases} \text{si } x<0, \text{alors } f(x)=-3x+1 \\ \text{si } x\geq 0, \text{alors } f(x)=3x+1 \end{cases}$</td>
<td>Avec Scratch

 
</td>
<td>Pseudo-code

Saisir x
Si x < 0 alors
y ← – 3 x + 1
Sinon
y ← 3 x + 1
Afficher y</td>
</tr>
<tr>
<td>Python sans fonction


```
x=float(input("Choisir un nombre x"))
if x<0:
    y=-3*x+1
else:
    y=3*x+1

print("L'image de x par f est ",y)
```

</td>
<td>Python et fonction


```
def f(x):
    if x<0:
        return -3*x+1
    else:
        return 3*x+1

t=float(input("Choisir un nombre x"))
print("L'image de x par f est ",f(t))
```

</td>
<td>Fonction Python


```
def f(x):
    if x<0:
        return -3*x+1
    else:
        return 3*x+1
```

</td>
</tr>
</table>

À vous de jouer !

Exercice à destination des élèves :

Un négociant en champagne envoie une publicité à ses clients pour les fêtes de Noël.

Habituellement, le prix d'une bouteille de champagne chez ce négociant est de 20 euros.

Il propose la bouteille à 17 euros si le nombre de bouteilles commandées est inférieur ou égal à 10 et dans ce cas la livraison est offerte. Par contre, si le nombre de bouteilles commandées est supérieur à 10 alors le prix d'une bouteille est de 13 euros et la livraison coûte 40 euros.

1. Quel prix paiera un client qui commande 8 bouteilles ? 25 bouteilles ?
2. Soit f la fonction qui, au nombre x de bouteilles commandées, associe le montant de la facture en euros.
   a. Quel est l'ensemble de définition de cette fonction .
   b. Donner l'écriture algébrique de cette fonction f.
3. a. Écrire, en pseudo-code, un algorithme affichant le montant de la facture en euros pour un nombre x de bouteilles commandées.
   b. Écrire cet algorithme en langage python sur ordinateur puis le tester.

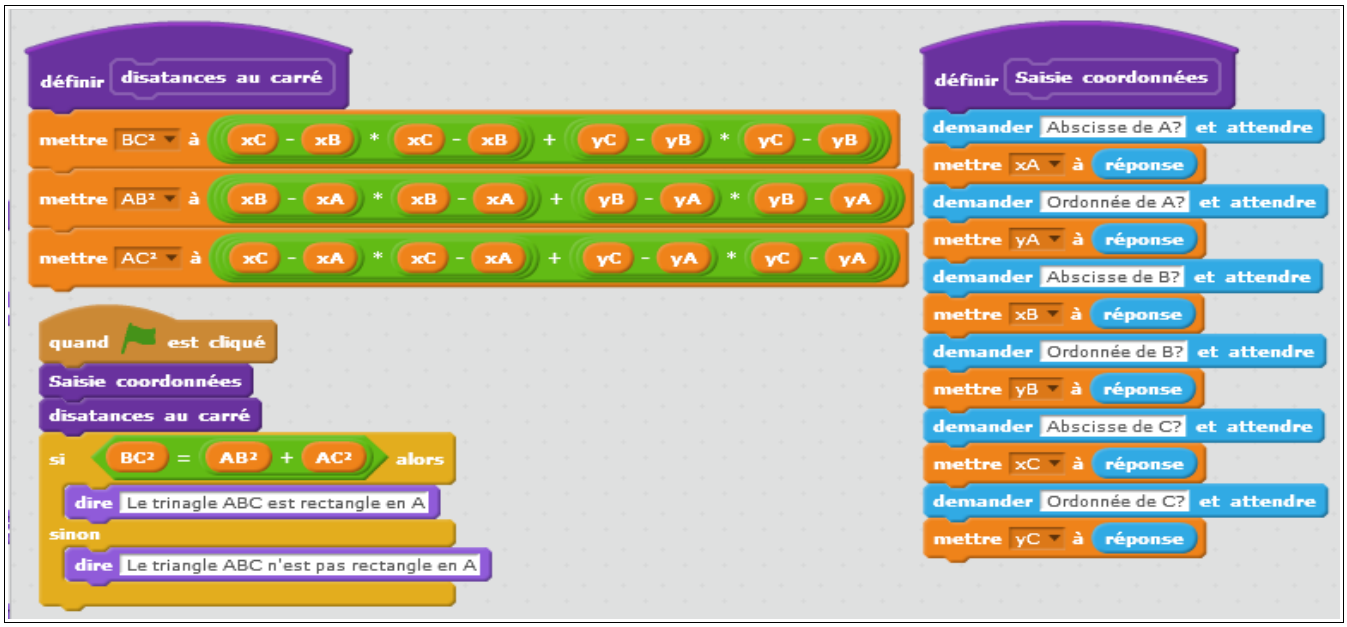
Travail pour les stagiares :

Réaliser la question **3. b**. uniquement.

Exemple de situation 3 : autour de la géomérie repérée.

Le plan muni d'un repère orthonormé. Soient trois points $A(xA;yA)$, $B(xB;yB)$, $C(xC;yC)$.

On a écrit le script Scratch ci-dessous qui permet de déterminer si le triangle ABC est rectangle en A.



… la saisie des formules de calcul de distances s'avère particulièrement fastidieuse avec Scratch !!!

Le langage Python facilite cette saisie grâce à l'utilisation de fonctions et du mode console.

<u>Pseudo-code</u>

Si $BC^2=AB^2+AC^2$ alors
    "Le tirangle ABC est rectangle en A"
Sinon
    "Le tirangle ABC n'est pas rectangle en A"

<u>Avec fonctions en Python</u>

```
def distance_carre(x,y,z,t):
    '''cette fonction renvoie la valeur du carre de la distance entre
    le point de coordonnees (x;y) et le point de coordonnees (z;t)'''
    return (z-x)**2+(t-y)**2

def test_rectangle_en_A(xA,yA,xB,yB,xC,yC):
    '''cette fonction teste si le triangle ABC est rectangle A lorsque
    les coordonnees des trois sommets sont rentrees dans cet ordre'''
    if distance_carre(xB,yB,xC,yC)==distance_carre(xA,yA,xB,yB)+distance_carre(xA,yA,xC,yC):
        return "le triangle ABC est rectangle en A"
    else:
        return "le triangle ABC n'est pas rectangle en A"
```

Commentaire : si la fonction « test_rectangle_en_A » est utilisée dans un programme on renverra plutôt les booléen « True » et « False » en sortie.

Exercice à destination des élèves :

1. Écrire, en pseudo-code, un algorithme qui permet de dire si le triangle ABC est rectangle en A, en B ou en C, ou s'il n'est pas rectangle.
2. En utilisant la fonction « distance_carre » définie dans le programme écrit en Python et en s'inspirant de ce programme, proposer une fonction Python qui permette de dire si un triangle ABC est rectangle ou non et si oui, en quel point.
3. Tester ce script en salle informatique.

Travail pour les stagiares :

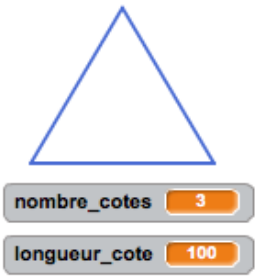
Écrire des fonctions Python qui utilisent les fonctions « distance_carre » et « test_rectangle_en_A » pour tester si un quadrilatère est un losange ou si un quadrilatère est un rectangle à partir des coordonnées de ses quatre sommets (Prolongement : écrire une fonction qui teste la nature d'un quadrilatère excepté le trapèze).

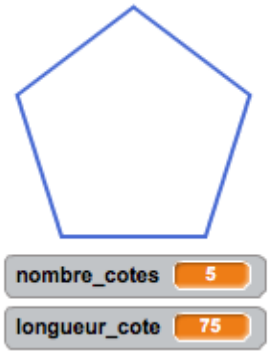
Les boucles

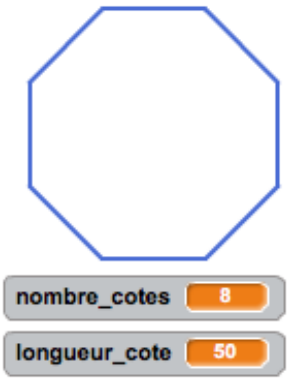
Exemple de situation 4 : tracés de figures géométrique.

Exercice à destination des élèves :

Créer un script Scratch permettant de tracer un polygone régulier où le nombre de côtés et la longueur d'un côté sont choisis par l'utilisateur en entrée.



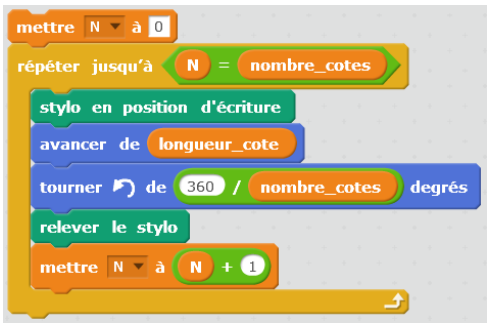




Institutionnalisation : écriture d'une boucle : Scratch – pseudo-code – Python.

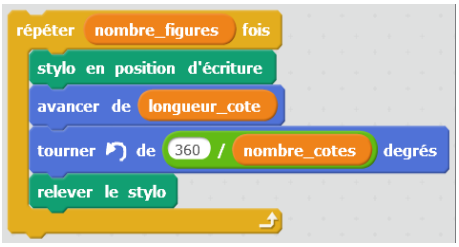
Boucle « while »

Script Scratch	Pseudo-code	Code Python Avec module turtle

	N ← 0
	Tant que N≠nombre_cotes
	stylo en position d'écriture
	avancer de longueur_cote
	tourner de
	360/nombre_cotes
	relever le stylo
	N ← N+1
	Fin tant que
		`N=0`
		`while (N!=nombre_cotes):`
		`down()`
		`forward(longueur_cote)`
		`left(360/nombre_cotes)`
		`up()`
		`N=N+1`

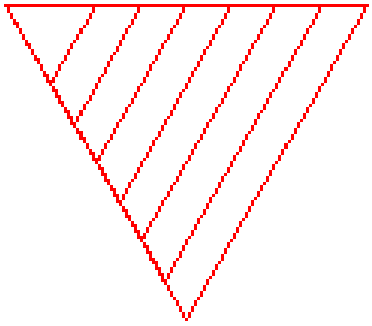
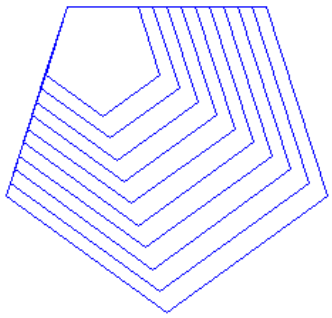
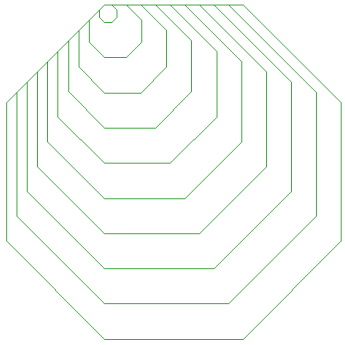
Boucle « for »

Script Scratch	Pseudo-code	Code Python Avec module turtle

	Pour N allant de 1 à nombre_cotes
	stylo en position d'écriture
	avancer de longueur_cote
	tourner de 360/nombre_cotes
	relever le stylo
	Fin Pour
		`for N in range(0,nombre_cotes):`
		`  down()`
		`  forward(longueur_cote)`
		`  left(360/nombre_cotes)`
		`  up()`

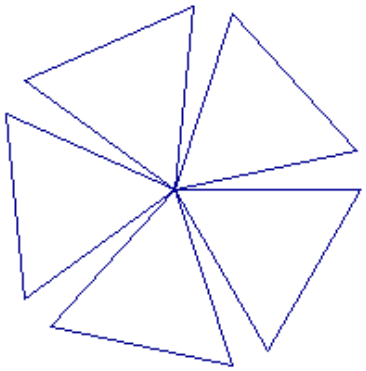
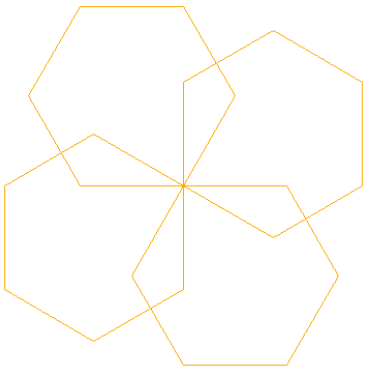
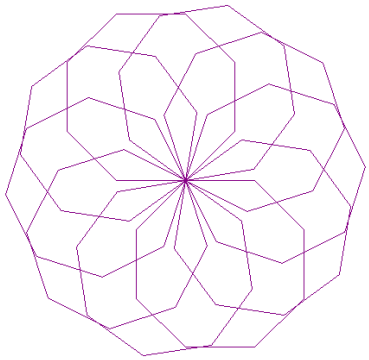
Travail pour les stagiaires (et les élèves ...):

1. Écrire une fonction « PolygoneRegulier » en Python qui prend en paramètres les variables « nombre_cotes » et « longueur_cote » qui dessine un polygone régulier à « nombre_cotes » côtés dont les longueurs mesurent « longueur_cote ».
2. **Défi 1 :**Écrire un code Python utilisant la fonction PolygoneRegulier permettant d'obtenir les figures suivantes avec des variables correspondant au nombre de côtés, à la longueur d'un côté, à la couleur, au nombre de figures et au pas.

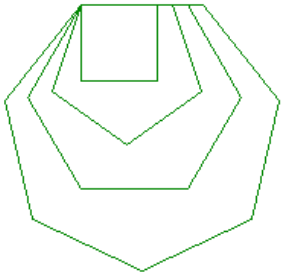
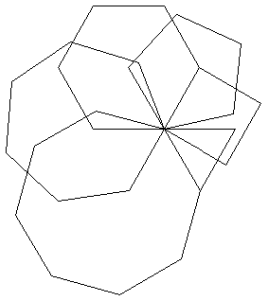
défi 1	 nombre de côtés : 3 longueur côté : 30 couleur : red nombre figures : 7 pas : 15	 nombre de côtés : 5 longueur côté : 25 couleur : blue nombre figures : 10 pas : 10	 Nombre de côtés: 8 longueur côté : 10 couleur : green nombre figures : 10 pas 20

Prolongement possible pour les stagiaires (… et les élèves?) :

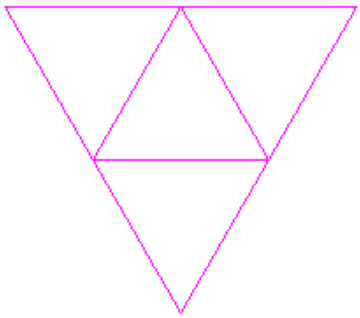
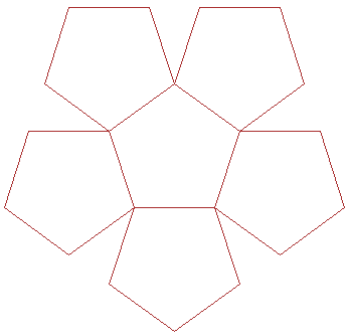
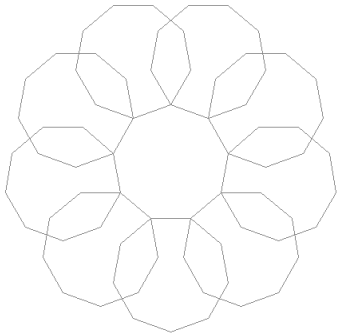
Défi 2 : Écrire un code Python utilisant la fonction PolygoneRegulier permettant d'obtenir les figures suivantes avec des variables correspondant au nombre de côtés, à la longueur d'un côté, à la couleur et au pas.

défi 2	 nombre de côtés : 3 longueur côté : 100 couleur : darkblue pas : 5	 nombre de côtés : 6 longueur côté : 100 couleur : orange pas : 4	 nombre de côtés : 8 longueur côté : 80 couleur : purple pas : 10

Défis 1bis et 2bis : Modifier les codes Python des défis 1 et 2 afin de pouvoir obtenir le figures suivantes.

défi 1bis	**défi 2bis**
nombre côtés : 4 longueur côté : 50 couleur :green nombre figures : 4 pas : 10 	nombre côtés : 3 longueur côté : 40 couleur :black pas : 6 

Défi 3 : Écrire un code Python utilisant la fonction PolygoneRegulier permettant d'obtenir les figures suivantes avec des variables correspondant au nombre de côtés, à la longueur d'un côté, à la couleur .

défi 3	 nombre côtés : 3 longueur côté : 100 couleur : magenta	 nombre côtés : 5 longueur côté : 80 couleur : brown	 nombre côtés : 9 longueur côté : 50 couleur : grey