ATELIER : De Scratch à Python
Corrections et prolongements

Les instructions d'affectation et l'utilisation de fonctions

Exemple de situation 1 : un programme de calcul.

Python sans fonction

```
x=float(input("Saisir un nombre: "))
a=4*x+3
b=2*x+5
y=a-b
print("Le nombre obtenu est : ",y)
```

Fonction Python

```
def f(x):
    a=4*x+3
    b=2*x+5
    return (a-b)
```

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

Travail pour les stagiares :

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

On privilégiera une solution avec fonction !

```
def facture(nb_bouteilles):
    if nb_bouteilles<10:
        return 17*nb_bouteilles
    else:
        return 13*nb_bouteilles+40
```

Exemple de situation 3 : autour de la géomérie repérée.

Les fonctions solutions ...

```
def distance_carre(x,y,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 en A en saisissant
    les coordonnees des 3 sommets dans cet ordre'''
    if distance_carre(xB,yB,xC,yC)==distance_carre(xB,yB,xA,yA)+distance_carre(xA,yA,xC,yC):
        return True
    else:
        return False

def milieu(x,y,z,t):
    abscisse=(x+z)/2
    ordonnee=(y+t)/2
    return(abscisse,ordonnee)

def parallelogramme(xA,yA,xB,yB,xC,yC,xD,yD):
    '''la fonction patallelogramme permet de dire si le quadrilatere ABCD est
    un parallelogramme en connaissant les coordonnees des 4 sommets rentrees
    dans cet ordre A, B, C, D '''
    if milieu(xA,yA,xC,yC)==milieu(xB,yB,xD,yD):
        return True
    else:
        return False

def rectangle(xA,yA,xB,yB,xC,yC,xD,yD):
    if parallelogramme(xA,yA,xB,yB,xC,yC,xD,yD)==True and test_rectangle_en_A(xA,yA,xB,yB,xD,yD)==True:
        return True
    else:
        return False
```

```python
def losange(xA,yA,xB,yB,xC,yC,xD,yD):
    if parallelogramme(xA,yA,xB,yB,xC,yC,xD,yD)==True and distance_carre(xA,yA,xB,yB)==distance_carre(xA,yA,xD,yD):
        return True
    else:
        return False

def carre(xA,yA,xB,yB,xC,yC,xD,yD):
    if losange(xA,yA,xB,yB,xC,yC,xD,yD)==True and rectangle(xA,yA,xB,yB,xC,yC,xD,yD)==True:
        return True
    else:
        return False

def quadrilatere(xA,yA,xB,yB,xC,yC,xD,yD):
    if parallelogramme(xA,yA,xB,yB,xC,yC,xD,yD)==True:
        print("ABCD est un parallelogramme")
    elif rectangle(xA,yA,xB,yB,xC,yC,xD,yD)==True:
        print("ABCD est un rectangle")
    elif losange(xA,yA,xB,yB,xC,yC,xD,yD)==True:
        print("ABCD est un losange")
    elif carre (xA,yA,xB,yB,xC,yC,xD,yD)==True:
        print("ABCD est un carre")
    else:
        print(" Le quadrilatere n'est ni un parallelogramme, ni un rectangle, ni un losange, ni un carré.")
```

Les boucles

Exemple de situation 4 : tracés de figures géométrique.

Script Scratch polygone régulier Fonction Python PolygoneRegulier	Fonction Python PolygoneRegulier
	


```python
from turtle import *

hideturtle()


def PolygoneRegulier(NbCote,TailleCote):
    down()
    for c in range(NbCote):
        forward(TailleCote)
        right(360/NbCote)
    up()
```

Défi 1 :

```python
from turtle import *

hideturtle()


def PolygoneRegulier(NbCote,TailleCote):
    down()
    for c in range(NbCote):
        forward(TailleCote)
        right(360/NbCote)
    up()

#Solution etape 3 (defi 1 et defi 1bis)
def Figure1(NC,TC,NF,CC,P):
    for i in range(NF):
        up()
        goto(-200,200)
        down()
        color(CC)
        PolygoneRegulier(NC,TC)
        up()
        #NC+=1 <-- Ã  rendre actif pour le defi 1bis
        TC=TC+P
```

Prolongements possibles boucles

Défi 2, 1bis et 2bis

```
from turtle import *

hideturtle()

def PolygoneRegulier(NbCote,TailleCote):
    down()
    for c in range(NbCote):
        forward(TailleCote)
        right(360/NbCote)
    up()

#Solution etape 3 (defi 2 et defi 2bis)
def Figure2(NC,TC,CC,P):
    angle=0
    while (angle<360):
        up()
        down()
        color(CC)
        setheading(angle)
        PolygoneRegulier(NC,TC)
        up()
        #NC+=1 <-- Ã  rendre actifpour le defi 2bis
        angle=angle+360//P
```

Défi 3

```
from turtle import *

hideturtle()

def PolygoneRegulier(NbCote,TailleCote):
    down()
    for c in range(NbCote):
        forward(TailleCote)
        right(360/NbCote)
    up()

#Solution etape 3 (defi 3)
def Figure3(NC,TC,CC):
    for i in range(NC):
        up()
        down()
        color(CC)
        PolygoneRegulier(NC,TC)
        up()
        forward(TC)
        left(360/NC)
```

Compléments

Un défi mathématique de calculs successifs

1. Faire tourner (en débranché) l'algorithme ci-dessous écrit à l'aide de scratch :

   On prendra comme valeur pour le nombre de départ 2 ; 1 ; 7 ; -1/3.

   Que se passe-t-il si on prend *x* en entrée ?

Sans fonction :

```
x=float(input("valeur en entrée ? "))
k=3*x-5
p=x+1
sortie=k/p
print("valeur en sortie ",sortie)
```

Avec fonction :

```
def calcul(x):
    return((3*x-5)/(x+1))
```

2. **a.** Faire tourner l'algorithme donné ci-dessous.
   b. Existe-t-il des valeurs pour lesquelles il ne fonctionne pas ? Si oui, lesquelles ?
   c. En le faisant tourner plusieurs fois avec diverses entrées, que remarquer ?
   d. Démontrer cette conjecture.

Sans fonction :

```
x=float(input("valeur en entrée ? "))
for i in range(0,4):
    k=3*x-5
    p=x+1
    x=k/p

sortie=x
print("valeur en sortie ",sortie)
```

Avec fonction :

```
def calcul(x):
    return((3*x-5)/(x+1))

def boucle_calcul(x):
    for i in range(0,4):
        x=calcul(x)
    return x
```

Exemple de situation 3 : autour de la géométrie repérée.

Réciproque du théorème de Pythagore optimisé

On pourrait commencer par demander d'identifier le côté le plus long donc l'éventuelle hypoténuse. Cela nous amène à faire créer par les élèves une fonction renvoyant le maximum de trois nombres.

```
def CarreDistance(x,y,z,t):
    return((z-x)**2+(t-y)**2)

def maximum(u,v,w):
    if (u>=v and u>=w):
        return u
    elif (v>=u and v>=w):
        return v
    else:
        return w

def ReciproquePythagore(xA,yA,xB,yB,xC,yC):
    A=CarreDistance(xB,yB,xC,yC)
    B=CarreDistance(xA,yA,xC,yC)
    C=CarreDistance(xA,yA,xB,yB)
    if(maximum(A,B,C)==A and A==B+C):
       print("Le triangle ABC est rectangle en A")
    elif(maximum(A,B,C)==B and B==A+C):
       print("Le triangle ABC est rectangle en B")
    elif(maximum(A,B,C)==C and C==A+B):
       print("Le triangle ABC est rectangle en C")
    else:
       print("Le triangle ABC n'est pas rectangle")
```

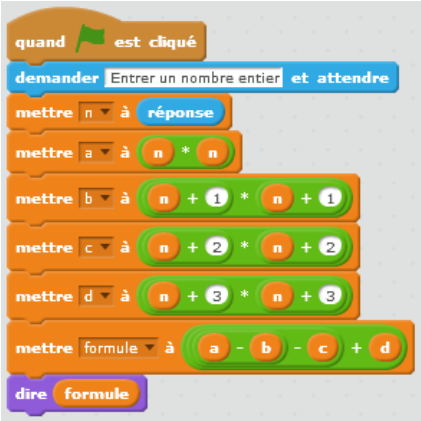
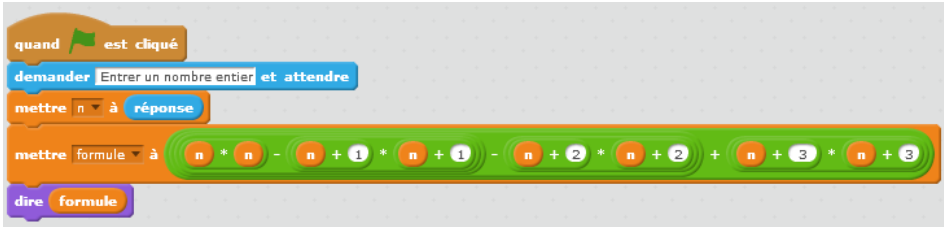
Une autre accorche possible pour un exemple de situation 1 : un programme de calcul.

Scénario activité élèves :

1. Ecrire un programme Scratch qui permet de caculer pour tout entier naturel *n* demandé en entrée à l'utilisateur de calculer la valeur de l'expression $n^2-(n+1)^2-(n+2)^2+(n+3)^2$.
   Le faire fonctionner pour plusieurs valeurs différentes de *n* en entrée.

→ ***Beaucoup d'élèves ont des résultats différents pour les mêmes valeurs en entrée ...***

2. On propose de passer à un autre langage de programmation, on donne aux élèves un code Python de ce programme de calcul et on leur demande de le saisir et le faire tourner. On fait le lien entre les deux en donnant le pseudo-code.
3. Faire plusieurs essais de calcul avec ce programme. Quelle conjecture pouvez-vous faire ?
4. Démontrez votre conjecture.
5. On donne aux élèves le code Python de ce programme de calcul avec une fonction et on leur demande de le saisir et le faire tourner. On constate qu'il fait la même chose.
   On institutionnalise la notion de fonction en Python. On en profite pour leur montrer l'utilisation du mode console pour les fonctions définies dans le code.

<table>
<tr>
<td>Scratch 4 afectations
 
</td>
<td>Scratch 1 affectation
 
</td>
</tr>
<tr>
<td>Fonction Python 4 affectations

```
def formule(x):
    a=x**2
    b=(x+1)**2
    c=(x+2)**2
    d=(x+3)**2
    return a-b-c+d
```

</td>
<td>Fonction Python 1 affectation

```
def formule(x):
    return x**2-(x+1)**2-(x+2)**2+(x+3)**2
```

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