INFO-H-100 - Programmation
TP 10 — Turtle

Pour les exercices suivants, écrivez vos programmes sur des fichiers avec IDLE et exécutez les. Veillez à découpez vos programmes en fonctions pertinentes.

Ex. 1. A l'aide des fonctions `fd` et `left` de `turtle`, écrire une fonction qui dessine un polygone régulier à n côtés de longueur l.

Ex. 2. A l'aide des fonctions `penup`, `pendown` et `goto` de `turtle`, écrire une fonction qui dessine le segment de droite entre les points (x_1, y_1) et (x_2, y_2).

Ex. 3. En utilisant la fonction `segment` de l'exercice 2, tapez le code suivant dans un script Python :

```
from math import pi, sin, cos

def dt(x1, y1, angle, niv):
    if niv > 0:
        x2 = x1 + int(cos(angle) * niv * 10.0)
        y2 = y1 + int(sin(angle) * niv * 10.0)
        segment((x1, y1), (x2, y2))
        dt(x2, y2, angle - pi/9, niv - 1)
        dt(x2, y2, angle + pi/9, niv - 1)

def drawTree(niv):
    dt(0, -200, pi/2, niv)

drawTree(9)
```

Comprenez-vous le comportement de la fonction `dt` ?

Ex. 4. A l'aide de l'exercice 2, écrire une fonction qui relie, dans l'ordre, tous les points d'une liste d'au minimum 2 points.

Ex. 5. A l'aide des exercices précédents et du tri par sélection, écrire une fonction qui relie tous les points d'une liste de points triés par ordre croissant en fonction de y.

Ex. 6. Ecrire une fonction qui produit les points de $100\sin(x/20)$ entre deux bornes $xMin$ et $xMax$ avec un pas donné.

```
calculPointsSin(-200, 200, 5) # -> [(-200, 54.40211108893698), (-195, 31.951919362227365), ...
```

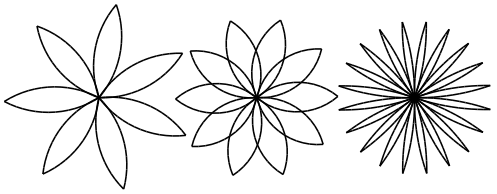
Ensuite, à l'aide des exercices précédents, dessiner cette liste.

Ex. 7. Ecrire une fonction qui produit les points d'une fonction $y = f(x)$ donnée en paramètres entre deux bornes $xMin$ et $xMax$ avec un pas donné.

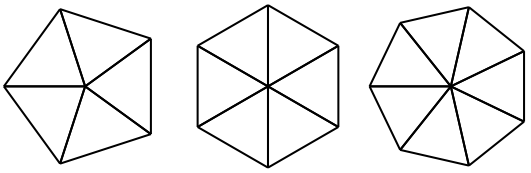
```
def f(x):
  return 100 * math.sin(x/20.0)

l = listeFonction(f, -200, 200, 5)

# -> [(-200, 54.40211108893698),
#     (-195, 31.951919362227365), ...
```

Ex. 8. A l'aide des exercices précédents, écrire une fonction qui dessine une fonction $y = f(x)$ agrandie de façon à ce que le dessin occupe tout l'espace entre $(-250, 250)$ et $(250, -250)$.

Ex. 9. Entrainez vous à dessiner des arcs de cercles à l'aide de la fonction `circle` de `turtle`. Ensuite, écrire une fonction qui dessine des fleurs de la manière suivante (conseil : dessiner des pétales) :



Ex. 10. Ecrire une fonction qui dessine des tartes de la manière suivante (conseil : dessiner des triangles isocèles) :



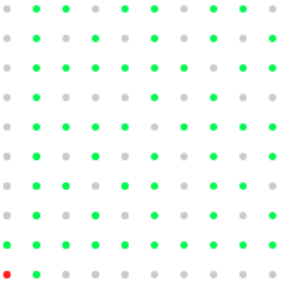
Ex. 11. A l'aide uniquement des fonctions `fd`, `left` et `setheading` (`turtle.setheading(0)` met à 0 l'angle de la tortue) de `turtle`, écrire une fonction qui dessine le segment de droite entre les points (x_1, y_1) et (x_2, y_2).

L'angle entre l'axe des ordonnées et la droite (x_1, y_1) et (x_2, y_2) est défini par

$$\arcsin\left(\frac{(y_2 - y_1)}{d((x_1, y_1), (x_2, y_2))}\right) * \frac{180}{\pi}$$

On considère que le point $(0, 0)$ est le point où la tortue commence son tracé (centre de l'écran).

Ex. 12. Dans le verger ci-dessous, seuls les arbres visibles de l'entrée du verger (le point rouge en bas à gauche), sont indiqués en vert. Les autres (les points gris) sont masqués par d'autres arbres. A l'aide de la fonction `dot`, `pencolor` et d'autres déjà vues, écrivez une fonction qui dessine un tel verger de coté `n` donné.



INFO-H-100 - Programmation
TP 10 — Turtle
Corrections

Solution de l'exercice 1:

```
import turtle

def polygone(n,l):
    angle = 360.0/n
    for i in range(n):
        turtle.fd(l)
        turtle.left(angle)

polygone(5,100)
polygone(12,50)
```

Solution de l'exercice 2:

```
import turtle

def segment(p1, p2):
    turtle.penup()
    turtle.goto(p1)
    turtle.pendown()
    turtle.goto(p2)

segment((0,0), (100,100))
segment((100,100),(200,0))
segment((200,0),(100,-100))
segment((100,-100),(0,0))
```

Solution de l'exercice 4:

```
def dessine(liste):
    if(len(liste)>=2):
        for i in range(1,len(liste)):
            segment(liste[i-1],liste[i])

l = [(-100,-100),(-100,100),(0,200),(100,100),
     (-100,100),(100,-100),(100,100),(-100,-100),
     (100,-100)]

dessine(l)
```

Solution de l'exercice 5:

```
def plusPetitPointY((x1,y1),(x2,y2)):
    return y1 < y2

def posMinFrom(ls, i, compare):
    """La position du min de ls en commencant en i.
    Pre: i dans ls
    """
    res = i
    while i < len(ls):
        if compare(ls[i],ls[res]):
            res = i
        i += 1
    return res

def echange(ls, i1, i2):
    ls[i1], ls[i2] = ls[i2], ls[i1]

def triSelection(ls, compare):
    for i in range(len(ls) - 1):
        pos = posMinFrom(ls, i, compare)
        echange(ls, i, pos)
```

```
l = [(20,10),(1,2),(40,50),(10,30),(100,2),(20,100),(45,76)]

triSelection(l,plusPetitPointY)
dessine(l)
```

Solution de l'exercice 6:

```
def calculPointsSin(xMin,xMax,pas):
    l = []
    for x in range(xMin,xMax+1,pas):
        l.append((x,100*math.sin(x/20.0)))
    return l

turtle.speed(0)
l= calculPointsSin(-200,200,5)
dessine(l)
```

Solution de l'exercice 7:

```
def f(x):
    return 100 * math.sin(x/20.0)

def listeFonction(f,xMin,xMax,pas):
    l = []
    for x in range(xMin,xMax+1,pas):
        l.append((x,f(x)))
    return l
```

Solution de l'exercice 8:

```
def minY(l):
    minimum = l[0][1]
    for point in l:
        if(point[1] < minimum):
            minimum = point[1]
    return minimum

def maxY(l):
    maximum = l[0][1]
    for point in l:
        if(point[1] > maximum):
            maximum = point[1]
    return maximum

def f2(x):
    return math.cos(x/10.0)

def dessineZoom(f,xMin,xMax):
    TAILLE = 250
    l = listeFonction(f,xMin,xMax,1)
    xRatio = float(2*TAILLE)/(abs(xMin)+abs(xMax))
    yMin = minY(l)
    yMax = maxY(l)
    yRatio = float(2*TAILLE)/(abs(yMin)+abs(yMax))
    for i in range(len(l)):
        l[i] = (l[i][0]* xRatio, l[i][1]*yRatio)
    xDepart = l[0][0]
    yDepart = l[0][1]

    dessine(l)

dessineZoom(f2,-100,100)
```

Solution de l'exercice 9:

```
import turtle

turtle.reset()

def petale(rayon, angle):
    for i in range(2):
        turtle.circle(rayon, angle)
        turtle.left(180-angle)
```

```
def fleur(nbPetales, rayon, angle):
    for i in range(nbPetales):
        petale(rayon,angle)
        turtle.left(360.0/nbPetales)

fleur(20,140,20)
```

Solution de l'exercice 10:

```
import turtle
import math

turtle.reset()

def tarte(n, r):
    theta = 360.0 / n
    for i in range(n):
        triangleIsocele(r, theta/2)
        turtle.left(theta)

def triangleIsocele(r, theta):
    y = r * math.sin(theta*math.pi/180)
    turtle.right(theta)
    turtle.fd(r)
    turtle.left(90+theta)
    turtle.fd(2*y)
    turtle.left(90+theta)
    turtle.fd(r)
    turtle.left(180-theta)

tarte(10,30)
```

Solution de l'exercice 11:

```
import turtle
import math

def distance((x1,y1),(x2,y2)):
    return math.sqrt((x1-x2)**2+(y1-y2)**2)

def angle((x1,y1),(x2,y2)):
    """pre: points != """
    sinus = (y2-y1)/distance((x1,y1),(x2,y2))
    angle = math.asin(sinus)*180/math.pi
    if(x1>=x2):                 #vers la gauche
        return - (angle + 180)
    else:                       #vers la droite
        return angle

def dessine_segment((x1,y1),(x2,y2)):
    turtle.setheading(0)
    d = distance((x1,y1),(x2,y2))
    a = angle((x1,y1),(x2,y2))
    turtle.left(a)
    turtle.fd(d)

dessine_segment((0,0), (100,100))
dessine_segment((100,100),(200,0))
dessine_segment((200,0),(100,-100))
dessine_segment((100,-100),(0,0))
```

Solution de l'exercice 12:

```
import turtle

def dessinePoint(p):
    turtle.penup()
    turtle.goto(p)
    turtle.dot()

def pgcd(a, b):
    if b == 0:
```

```
        return a
    else:
        return pgcd(b, a % b)

def verger(n):
    turtle.hideturtle()
    for l in range(n):
        for c in range(n):
            if c == 0 and l == 0:
                turtle.pencolor("red")
            if pgcd(l, c) > 1:
                turtle.pencolor("grey")
            dessinePoint((20 * c , 20 * l))
            turtle.pencolor("green")

verger(10)
```