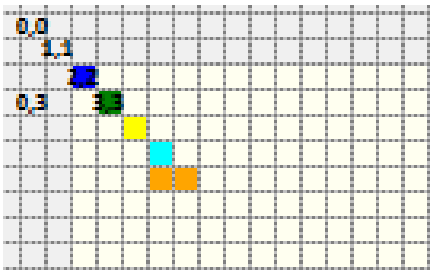
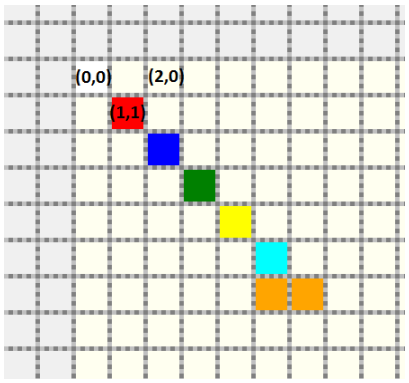
FICHE PYTHON 14 :
INTERFACE GRAPHIQUE 3 - CANVAS (4 pages)

Bibliothèque à importer pour créer des applications graphiques	`#!/usr/bin/env python` `# -*- coding: utf-8 -*-` `from tkinter import *`
Création d'une fenêtre	`fen_princ = Tk()` `# Insérer ici vos différents widgets` `fen_princ.mainloop()`
Modification du titre et de la taille de la fenêtre	`fenetre = Tk()` `fenetre.title("NOM DE VOTRE PROGRAMME")` `fenetre.geometry("500x500")`
Destruction d'une fenêtre	`fenetre.destroy()`

Création et gestion du canvas

Création d'un Canvas	`monCanvas = Canvas(fen_princ, width=500, height=500, bg='ivory')` Création d'un canvas avec une position (0,0) qui inclut une marge externe non dessinable de 2 pixels. 
Création d'un Canvas **Sans bordure**	`monCanvas = Canvas(fen_princ, width=499, height=499, bg='ivory', borderwidth=0, highlightthickness=0)` 

Dessiner des lignes

Méthode
create_line()

```
monCanvas.create_line(x1, y1, x2, y2, width=2, fill="red")
monCanvas.create_line(x1, y1, x2, y2, width=2, fill="#ff0000")
```

Permet de créer une ligne rouge :
- partant du point (x1,y1) inclus et
- allant au point (x2,y2) exclus
- avec une épaisseur de 2 pixels.

Le paramètre fill correspond à la couleur du trait, qu'on peut fournir sous différents formats, donc le RGB en hexadécimal.

Fonction permettant de dessiner des lignes en donnant :
- *le point de départ (x0,y0)*
- *la longueur*
- *l'angle en degré*
- *la couleur*

```
def rotation_ligne(leCanvas, x0, y0, longueur, angle, couleur):
    # x0, y0 sont les coordonnées d'origine
    # longueur est la longueur de la ligne en pixels
    # couleur est la couleur sous forme de string : "red" ou "#ff0000"
    # angle est l'angle par rapport à l'horizontal en degré
    angle = math.radians(angle)
    # math.radians permet de convertir en radians
    xf=int(x0+longueur*math.cos(angle))
    yf=int(y0+longueur*math.sin(angle))
    leCanvas.create_line(x0, y0, xf, yf, width=2, fill=couleur)
```

Exemple d'appel depuis le programme qui inclut le module math :

```
import math
rotation_ligne(monCanvas, 250, 250, 100, 45, "#ff0000")
```

Dessiner des rectangles

Méthode
create_rectangle()

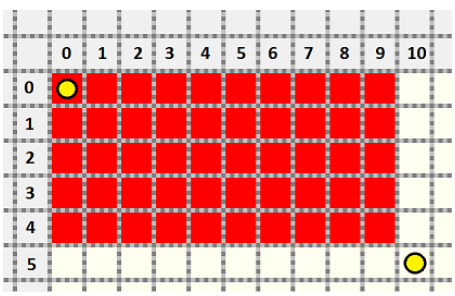
```
monCanvas.create_rectangle(100,200,300,400, fill="red",
activefill="yellow", outline="blue", width=5)
```

- Obligatoire : Coordonnées du coin supérieur gauche (intérieur au rectangle) : x0,y0.
- Obligatoire : Coordonnées du coin inférieur droit (extérieur au rectange) : x1,y1.
- Optionnel : Couleur de remplissage avec **fill**. Transparent par défaut.
- Optionnel : Couleur de remplissage lorsqu'on survole l'objet avec la souris avec **activefill** (cela fonctionne également avec les lignes d'ailleurs).
- Optionnel : Couleur de contour avec **outline**. "black" par défaut.
- Optionnel : Largeur du contour avec **width**. 1.0 par défaut.

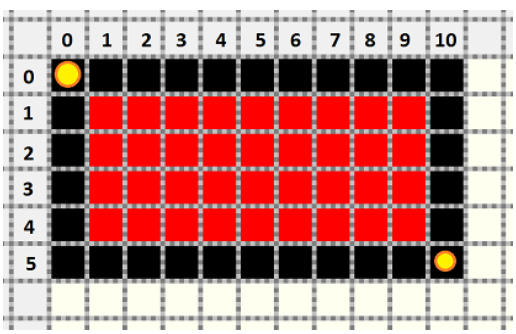
Attention : Attention à la coordonnée finale : il s'agit du pixel juste extérieur à la boîte interne. Ca fonctionne « normalement » avec une épaisseur de 1 par contre. Si vous ne voulez pas vous compliquer la vie, vous pouvez créer des rectangles d'épaisseur 1 pixel avec la même couleur pour fill et outline.

ATTENTION : la gestion des rectangles ayant des bordures importantes est assez compliquée. Voir l'activité elle-même.

```
monCanvas.create_rectangle(0,0,10,5
,width=0,fill="red")
```




```
monCanvas.create_rectangle(0,0,10
,5,width=1,fill="red")
```



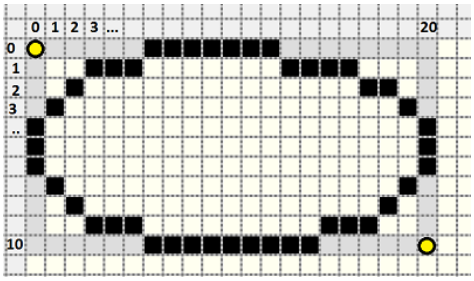
Dessiner des ellipses

Méthode
create_oval()

Pour décrire une ellipse avec Tkinter, il faut trouver le rectangle d'épaisseur qui la contient. On a alors simplement besoin des coordonnées (x_0,y_0) du point supérieur gauche et des coordonnées (x_1,y_1) du point inférieur droit du rectangle.

```
monCanvas.create_oval(0,0,20,10)
```

Crée une ellipse de bordure 1 pixel qu'on peut dessiner dans le rectangle d'épaisseur 1 pixel dessinable entre (0;0) et (20;10).



Autres paramètres

On trouve beaucoup d'arguments permettant de personnaliser les ellipses, comme pour les lignes et les rectangles. Parmi celles que je vous propose ici :

- Pour remplir la forme avec une couleur : `fill="#445566"` ou `fill="blue"`, transparent par défaut.
- Pour modifier la taille de la bordure : `width=1` est la valeur par défaut.
- Pour modifier la couleur de la bordure : `outline="#dddddd"` par exemple, "black" par défaut.
- Pour changer la couleur de remplissage au survol : `activefill="red"` par exemple.
- Pour changer la taille de la bordure au survol : `activewidth=4` par exemple.

Dessiner des polygones

Méthode
create_polygon()

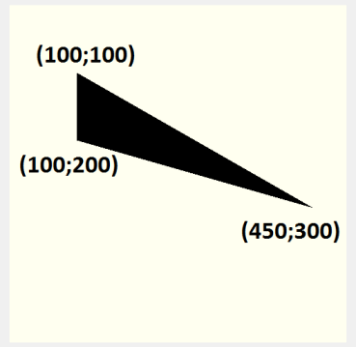
Pour définir un polygone, il faut au moins trois points non alignés.
Pour écrire un polygone dans Tkinter, on utilise

```
create_polygon(x0, y0, x1, y1, x2, y2 ...)
```

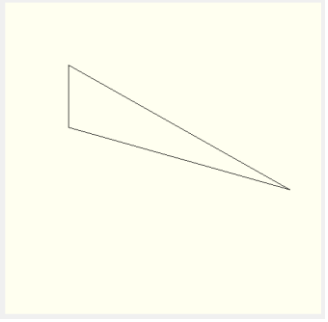
Vous pouvez utiliser les paramètres habituels pour gérer la largeur de la bordure (width) , le remplissage (fill) ...

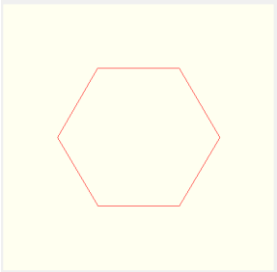
Les options par défaut sont : **fill="black"** (remplissage noir) et **outline=""** (bordure transparente).

```
monCanvas.create_polygon(100,100,100,200,450,300)
```

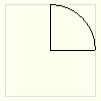
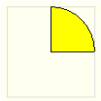
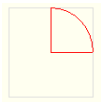
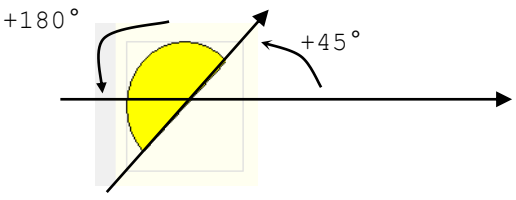



```
monCanvas.create_polygon(100,100,100,200,450,300, fill="", outline="black")
```



Création d'un hexagone en utilisant la fonction rotation_ligne 	``` def hexagone(leCanvas, x0,y0,longueur, couleur): # x0, y0 sont les coordonnées d'origine # longueur est la longueur entre le centre et les points extrêmes # couleur est la couleur sous forme de string : "red" ou "#ff0000" x=[] y=[] for i in range(6): angle = math.radians(i*360/6) x.append(int(x0+longueur*math.cos(angle))) y.append(int(y0+longueur*math.sin(angle))) leCanvas.create_polygon(x[0],y[0],x[1],y[1],x[2],y[2],x[3],y[3], x[4],y[4],x[5],y[5], fill="", outline="black") ``` En utilisant : `hexagone(monCanvas,250,250,150,"red")`, on obtient la figure ci-contre.

Dessiner des arcs d'ellipse ou de cercle

Méthode ***create_arc()***	On utilise la méthode **create_arc** en donnant : • le point supérieur gauche (qu'on nommera x0,y0) et • le point inférieur droit (qu'on nommera x1,y1) • du rectangle d'épaisseur 1 qui entoure l'arc de cercle ou d'ellipse.
	Exemple 1 : ``` monCanvas.create_rectangle(10,10,100,100,outline="#dddddd") monCanvas.create_arc(10,10,100,100) ```
	Exemple 2 : ``` monCanvas.create_rectangle(10,110,100,210,outline="#dddddd") monCanvas.create_arc(10,110,100,210, fill="yellow") ```
	Exemple 3 : ``` monCanvas.create_rectangle(10,220,100,320,outline="#dddddd") monCanvas.create_arc(10,220,100,320, outline="#ff0000") ```
Création d'un arc différent de 90° avec **start** et **extent**	Le paramètre **start** contient l'angle du trait initial par rapport à l'horizontal. Par défaut, `start = 0` (degré). Le paramètre **extent** contient l'angle de l'arc à créer. Par défaut `extend = 90` (degrés). On peut ainsi obtenir avec un start de 45° et un extend de 180° : ``` monCanvas.create_arc(10,110,100,210, fill="yellow",start=45,extent=180) ``` 

On parle également dans cette partie du module random et math.
Chercher dans l'activité s'il le faut.