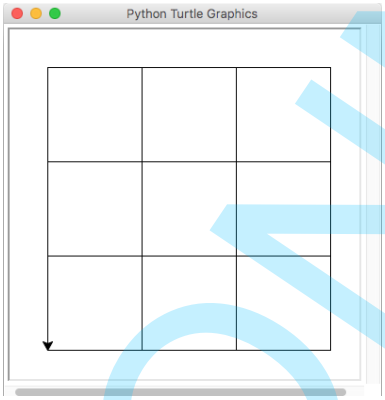
# Créer un premier jeu grâce au module Turtle de Python

Pour l'ensemble de cette activité, il est possible de se référer à la vidéo https://vimeo.com/267076019/c003c6321b.
L'ensemble des programmes écrits en Python utilise le module Turtle (*from turtle import **).
Importer également la bibliothèque Math lorsque la fonction racine carré (*sqrt()*) est nécessaire (*from math import **).
Vous pourrez utiliser la fonction *setheading(d)* qui attribue à la tortue un angle absolu de $d$ degrés par rapport à l'horizontale.

## I Premiers éléments graphiques

1. Tracer à l'aide d'un programme permettant de tracer un cadre de jeu (étape 1) dans une fenêtre 400 pixels × 400 pixels (écart entre chaque ligne : 100 pixels).




Ecrire le programme sous la forme :

```
from turtle import *

setup(400,400)#Fixer une fenêtre de 400x400

def cadrejeu():#Tracé du cadre de jeu
    ...


    ...


cadrejeu()
```

2. Ecrire une fonction *cercle(rayon)* permettant de tracer un cercle dont le centre est la position du curseur et le rayon est donné (étape 2). Utiliser la fonction *circle(rayon)* du module Turtle qui trace un point dont le « point de départ » est la position initiale du curseur.

```
from turtle import *

setup(400,400)#Fixer une fenêtre de 400x400

def cercle(rayon):#tracé d'un cercle de centre et de rayon donnés
    ...


    ...


cadrejeu()
```

3. Ecrire une fonction *croix(l)* permettant de tracer une croix dont le centre est la position du curseur et dont la largeur et la hauteur sont de $2l$ pixels (étape 3).

# II Création de plusieurs tortues

1. Exécuter le programme suivant (étape 4) :

```
from turtle import *

setup(400,400)#Fixer une fenêtre de 400x400
title("Deux nouvelles tortues")
bgcolor("black")

def deuxtortues():#création de deux nouvelles tortues
        premiere=Turtle()
        seconde=Turtle()
        premiere.color("red")
        seconde.color("blue")
        premiere.up()
        premiere.goto(-100,-100)
        premiere.down()
        premiere.forward(250)
        premiere.left(90)
        seconde.circle(80)
        premiere.forward(200)
        seconde.right(90)
        seconde.forward(100)

deuxtortues()
```

2. Ecrire un programme correspondant à l'étape 5 de la vidéo.

# III Interactions à travers la souris

1. Observer et exécuter le programme suivant (étape 6) :

```
from turtle import *

setup(400,400)

hideturtle()#cache la tortue
speed(10)#augmente la vitesse à 10
seconde=Turtle()#création d'une nouvelle tortue
seconde.hideturtle()#cache la seconde tortue
seconde.speed(10)#augmente la vitesse de la seconde tortue à 10

def cercle(rayon):#tracé d'un cercle de centre et de rayon donnés
        up()
        left(90)
        down()
        circle(rayon)
        right(90)

def clicdanscercle(x,y):#analyse si le points de coordonnées (x,y) est dans le disque
        seconde.clear()
        seconde.up()
        seconde.goto(-50,150)
        seconde.color("red")
        if (x)**2+(y)**2<=100**2:
                seconde.write("Dedans", font = ("Arial", 32, "bold"))
        else:
                seconde.write("Dehors", font = ("Arial", 32, "bold"))

cercle(100)#tracé du cercle
onscreenclick(clicdanscercle)#en cas d'appui sur un bouton de la souris
#renvoie les coordonnées du clic de la souris dans la fonction clicdanscercle
mainloop()#indique la boucle d'écoute des événements (clavier ou souris)
```

2. Ecrire un programme permettant d'attribuer à la position d'un clic de la souris relativement à une cible, un score conformément à l'étape 7 de la vidéo.
   Vous pourrez utiliser une instruction conditionnelle :

```
...

if ... :
        ...

elif ... :
        ...

elif ... :
        ...

elif ... :
        ...

else ... :
        ...

...
```

3. Améliorer le programme précédent en indiquant le score total (étape 8).
   Pour cela initialiser une variable globale *totalpoints=0* qu'il est possible de modifier à l'intérieur d'une fonction en indiquant en première ligne de la fonction *global totalpoints* .

# IV Et le jeu dans tout cela ...

1. Ecrire un programme permettant de tracer alternativement des cercles et des croix centrés sur les clics successifs de la souris et de dimensions 80 pixels × 80 pixels (étape 9). Vous pourrez utiliser la présentation suivante :

```
from turtle import *
from math import *

setup(400,400)#Fixer une fenêtre de 400x400

def cadrejeu():#Tracé du cadre de jeu
        ...

def cercle(rayon):#tracé d'un cercle de centre et de rayon donnés
        ...

def croix(l):#tracé d'une croix de centre donné et de largeur 2l
        ...

hideturtle()#cache la tortue
speed(10)#augmente la vitesse à 10
seconde=Turtle()#création d'une nouvelle tortue
seconde.hideturtle()#cache la seconde tortue
seconde.speed(10)#augmente la vitesse de la seconde tortue à 10

ncoup=0#compte le nombre de coups joués

def morpion1(x,y):#trace en alternance un cercle et une croix centrés sur les clics successifs
        global ncoup
        ...

onscreenclick(morpion1)
mainloop()
```

2. Pour recentrer les tracés, on introduit la fonction :

```
def calcul(x) :
        return 100*((x+50)//100)
```

Ecrire un programme correspondant à l'étape 10 (ne pas oublier d'exclure les clics situés à l'extérieur de la grille).

3. Pour la suite, nous allons ajouter la bibliothèque NumPy (*from numpy import **).
Cette bibliothèque permet notamment une utilisation facilitées des matrices et des tableaux de valeurs.
Nous allons introduire la matrice $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ grâce à l'instruction :

```
tabl=array([[0,0,0],[0,0,0],[0,0,0]])
```

*tabl[i,j]* désigne le terme du tableau situé à l'intersection de la ligne $i$ et de la colonne $j$ de *tabl*. Ainsi *tabl[2,1]* désigne le terme situé à l'intersection de la deuxième ligne et de la première colonne de *tabl*.

Ci-dessous est écrit la fin du programme correspondant à l'étape 11.

```
...


tabl=array([[0,0,0],[0,0,0],[0,0,0]])#crée un tableau de 3 lignes et 3 colonnes ne contenant que des 0

def calcul2(x):#1
        return int((x+50)//100+1)

def morpion3(x,y):#2
        global tabl, ncoup
        if x>-150 and y>-150 and x<150 and y<150:
                xnum=calcul2(x)
                ynum=calcul2(y)
                if tabl[xnum,ynum]==0:#3
                        up()
                        x=calcul(x)
                        y=calcul(y)
                        goto(x,y)
                        if ncoup%2==0:
                                cercle(40)
                                tabl[xnum,ynum]=1#la cellule est complétée par le joueur 1
                        else:
                                croix(40)
                                tabl[xnum,ynum]=-1#la cellule est complétée par le joueur 2
                        ncoup+=1

cadrejeu()
onscreenclick(morpion3)
mainloop()
```

Décrire le rôle de la fonction *calcul2()* et compléter les commentaires #1, #2 et #3.
A quoi correspond *tabl[xnum,ynum]* ?

4. Ecrire un programme correspondant à l'étape 12 (vérification et affichage du résultat).
   Pour cela, vous pourrez introduire et compléter la fonction suivante :

```
def verification(tableau,n):#fonction permettant de vérifier si la partie est gagnée ou nulle
    if tableau[2,0]==1 and tableau[1,1]==1 and tableau[0,2]==1:
        vainqueur(1)
    if tableau[2,0]==-1 and tableau[1,1]==-1 and tableau[0,2]==-1:
        vainqueur(-1)
    if tableau[0,0]==1 and tableau[1,1]==1 and tableau[2,2]==1:
        vainqueur(1)
    if tableau[0,0]==-1 and tableau[1,1]==-1 and tableau[2,2]==-1:
        vainqueur(-1)
    for k in range(3):
        if tableau[k,0]==1 and tableau[k,1]==1 and tableau[k,2]==1:
            vainqueur(1)
            if tableau[k,0]==-1 and tableau[k,1]==-1 and tableau[k,2]==-1:
        vainqueur(-1)
        ...

    if n==9:
        vainqueur(0)
```

   Vous devrez créer une fonction *vainqueur(v)* écrivant à l'écran l'issue de la partie selon la valeur $v$ du vainqueur.

   Remarque : nous avons utiliser également une variable de test pour empêcher de relancer un tracé avant la fin du tracé précédent (facultatif).

5. Ecrire un programme correspondant à l'étape 13.
   Vous pourrez :
   - utiliser l'instruction *textinput("Rejouer ?", "Rejouer ? Veuillez entrer 'oui' si c'est le cas.")* renvoyant une chaîne de caractère entrée par l'utilisateur et pour laquelle on veut savoir si elle est égale à 'oui'.
   - utiliser la fonction *bye()* fermant la fenêtre Turtle.
   - utiliser la fonction *sys.exit(0)* (*import sys*) permettant de clore le programme.

6. L'étape 14 (jeu contre l'ordinateur) est un vrai défi. Il est possible d'utiliser la bibliothèque Random (*import random*) et l'instruction *random.randint(0,2)*.

7. Maxi défi : adapter ce qui précède pour créer un jeu type « Puissance 4 ».