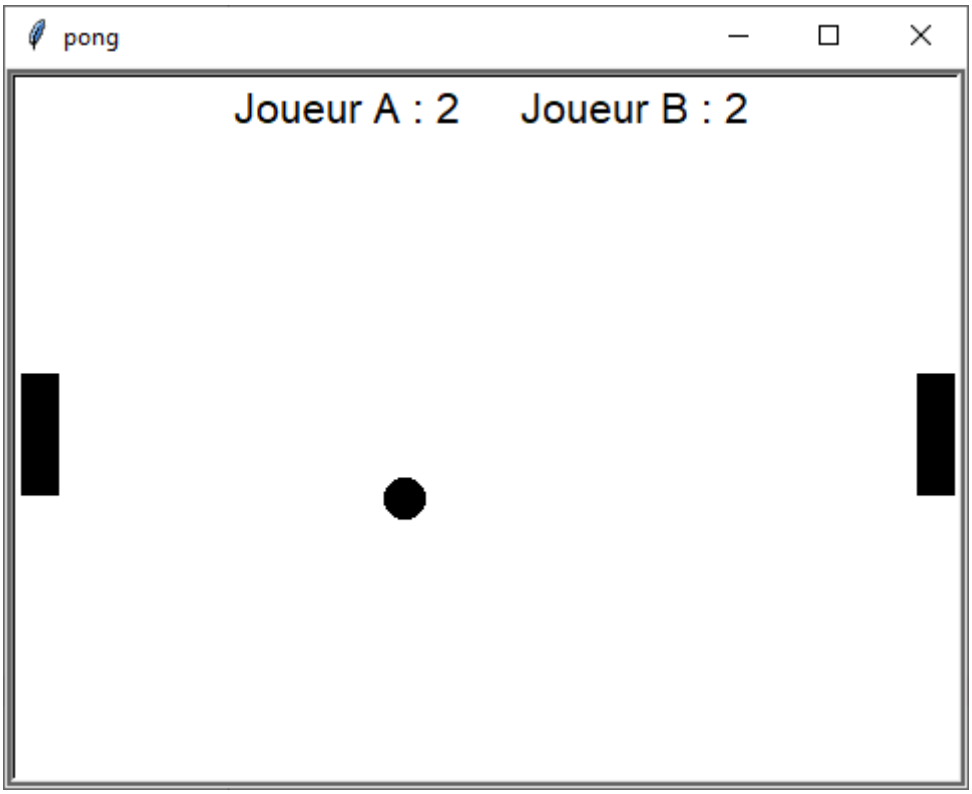
Jeu de Pong : Python-Turtle

Nous allons réaliser un jeu de Ping-Pong avec 2 joueurs : la balle peut rebondir sur les 4 bords du terrain. Si elle touche les bords à droite ou à gauche, le score d'un des joueurs est augmenté de 1. Nous programmons ce jeu avec le module turtle.



import turtle as tu

#---- définitions des fonctions utilisées ----

…

#---- définitions de l'interface et des objets de l'application

…

ecran.mainloop()

tu.bye()

Création de la fenêtre de jeu

Nous allons donner à l'écran les mêmes dimensions qu'une scène Scratch.

```
ecran=tu.Screen()

ecran.title("pong")

ecran.bgcolor("white")

ecran.setup(width=480, height=360)
```

Les objets :

2 raquettes rectangulaires 18 x 60 pixels.

Nous devons créer une forme rectangulaire qui servira ensuite comme forme d'un objet turtle.

```
#création d'une forme raquette

poly = ((0,0), (0,18),(60,18),(60,0))

name = "raquette"

a = tu.Shape("compound")

a.addcomponent(poly, "black")

tu.addshape(name, a)
```

Nous disposons maintenant d'une forme pour la tortue dont l'origine est en haut à gauche.

```
#Raquette A

raquetteA=tu.Turtle()

raquetteA.speed(0)

raquetteA.shape("raquette")

raquetteA.color("black")

raquetteA.penup()

raquetteA.goto(-235,30)
```

```
#Raquette B
raquetteB=tu.Turtle()
raquetteB.speed(0)
raquetteB.shape("raquette")
raquetteB.color("black")
raquetteB.penup()
raquetteB.goto(212,30) #normalement la valeur en x devrait être 235-18 = 217,
                       #mais sur mon écran cette valeur envoie la raquette
                       # trop à droite ??
```

La forme « circle » prédéfinie crée une tortue circulaire de 20 pixels de diamètre. Point de référence au centre du cercle.

```
#Balle
balle=tu.Turtle()
balle.speed(0)
balle.shape("circle")
balle.color("black")
balle.penup()
balle.goto(randint(-220,220),randint(-160,160))
Balledx=2
Balledy=2
```

Les variables pour les scores et le crayon pour afficher ces scores :

```
#score
scoreA = 0
scoreB = 0
```

```
#un crayon pour afficher les scores
pen=tu.Turtle()
pen.speed(0)
pen.color("black")
pen.penup()
pen.hideturtle()
pen.goto(0,150)
pen.write("Joueur A : {}     Joueur B : {}".format(scoreA, scoreB),
align="center", font=("Arial",16,"normal"))
```

La gestion des raquettes

La raquette de gauche sera déplacée vers le haut ou le bas par les touches a et w

La raquette de droite sera déplacée vers le haut ou le bas par les touches flèche haut et flèche bas.

```
#lier l'écran aux évènements clavier pour déplacer les raquettes
ecran.listen()
ecran.onkeypress(raquetteA_up,"a")
ecran.onkeypress(raquetteA_down,"w")
ecran.onkeypress(raquetteB_up,"Up")
ecran.onkeypress(raquetteB_down,"Down")
```

Les fonctions appelées lorsque l'une de ces touches sera appuyée.

```
def raquetteA_up():
    y=raquetteA.ycor()
    y +=20
    raquetteA.sety(y)
```

```
def raquetteA_down():
    y=raquetteA.ycor()
    y -=20
    raquetteA.sety(y)

def raquetteB_up():
    y=raquetteB.ycor()
    y +=20
    raquetteB.sety(y)

def raquetteB_down():
    y=raquetteB.ycor()
    y -=20
    raquetteB.sety(y)
```

La boucle de jeu

Une fois les objets créés et placés sur l'écran, le jeu s'inscrit dans une boucle infinie, qui ne s'arrête que si on ferme la fenêtre.

Tester si un bord est touché

La balle se déplace en modifiant ses coordonnées d'une valeur dx pour les abscisses et de dy pour les ordonnées. Nous avons mémorisé ces valeurs dans 2 variables Balledx et Balledy : plus la valeur est grande et plus la balle se déplacera vite.

Lorsque la balle touche un bord ou une raquette, le signe de dx ou de dy est inversé pour que la balle reparte en sens inverse.

Si la balle touche un bord à droite ou à gauche, le joueur en face marque un point.

```
while True:
    ecran.update() #rafraichir l'écran à chaque tour
    #Déplacer la balle
    balle.setx(balle.xcor()+Balledx)
```

```
        balle.sety(balle.ycor()+Balledy)
        #tester les bords
        if balle.ycor()> 170:
            balle.sety(170)
            Balledy*=-1

        if balle.ycor()< -170:
            balle.sety(-170)
            Balledy*=-1

       if balle.xcor()> 230:
            balle.setx(230)
            Balledx *=-1
            scoreA +=1
            pen.clear()
            pen.write("Joueur A : {}     Joueur B : {}".format(scoreA, scoreB),
align="center",font=("Arial",16,"normal"))

      if balle.xcor()< -230:
            balle.setx(-230)
            Balledx *=-1
            scoreB +=1
            pen.clear()
            pen.write("Joueur A : {}     Joueur B : {}".format(scoreA, scoreB),
align="center",font=("Arial",16,"normal"))
```

Tester si une des raquettes est touchée

Lorsque la balle touche la raquette de gauche, son abscisse est inférieure ou égale à

–(235-18-10)=-207 et son ordonnée comprise entre (ordonnée de la raquetteA) et (ordonnée de la raquetteA - 60). Les raquettes ont 60 pixels de haut et le point de référence en haut à gauche.

Lorsque la balle touche la raquette de droite, son abscisse est supérieure ou égale à

212-10 = 202 et son ordonnée comprise entre (ordonnée de la raquetteB) et (ordonnée de la raquetteB - 60).

```
#test collisions balle - raquette

    if (balle.xcor()> 202) and (balle.ycor()< raquetteB.ycor() and
balle.ycor()>raquetteB.ycor()-60 ):

        balle.setx(200)

        Balledx *=-1


    if (balle.xcor()< -207) and (balle.ycor()< raquetteA.ycor() and
balle.ycor()>raquetteA.ycor()-60 ):

        balle.setx(-205)

        Balledx *=-1
```

Ajouter des sons lorsque la balle touche un bord ou une raquette

J'ai exporté les sons pop et snap de Scratch et obtenu 2 fichiers .wav

Pour faire jouer des sons à python, il faut :

- Importer winsound : **import winsound**
- Écrire : **winsound.PlaySound("pop.wav",winsound.SND_ASYNC)** pour faire jouer le son pop.
- **winsound.PlaySound("snap.wav",winsound.SND_ASYNC)** pour faire jouer le son snap

Les 2 fichiers doivent être dans le même dossier que le programme.

Une astuce

Lorsqu'on met au point un programme python, il arrive souvent que le programme plante et c'est un peu long d'attendre que Windows dise : ce programme ne répond pas etc.

On peut éviter cela en important le module traceback : **import traceback** et en englobant tout le code dans :

```
import traceback

try :

        tout notre code après les import et jusqu'à

        ecran.mainloop()

except:

        #ce bloc permet de récupérer des infos en cas d'erreur

         traceback.print_exc()

finally:

        tu.bye()
```