In [2]:

```
import turtle
```

In [3]:

```
wn = turtle.Screen()
wn.title("Pong by me")
wn.bgcolor("green")
wn.setup(width=800, height=600)
wn.tracer(0)
```

In [4]:

```
# Score
score_a = 0
score_b = 0
```

In [5]:

```
#Paddle A
paddle_a = turtle.Turtle()
paddle_a.speed(0)
paddle_a.shape("square")
paddle_a.color("white")
paddle_a.shapesize(stretch_wid=5 , stretch_len=1)
paddle_a.penup()
paddle_a.goto(-355,0)
```

In [6]:

```
#Paddle B
paddle_b = turtle.Turtle()
paddle_b.speed(0)
paddle_b.shape("square")
paddle_b.color("white")
paddle_b.shapesize(stretch_wid=5 , stretch_len=1)
paddle_b.penup()
paddle_b.goto(355,0)
```

In [7]:

```
#ball
ball = turtle.Turtle()
ball.speed(0)
ball.shape("square")
ball.color("white")
ball.penup()
ball.goto(0,0)
ball.dx=0.1
ball.dy=0.1
```

In [8]:

```
# Pen
pen = turtle.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 260)
pen.write("Player A: 0  Player B: 0", align="center", font=("Courier", 24, "normal"))
```

In [9]:

```
#function
def paddle_a_up():
    y = paddle_a.ycor()
    y += 20
    paddle_a.sety(y)

```

In [10]:

```python
#function
def paddle_a_down():
    y = paddle_a.ycor()
    y -= 20
    paddle_a.sety(y)
```

In [11]:

```python
#function
def paddle_b_up():
    y = paddle_b.ycor()
    y += 20
    paddle_b.sety(y)
```

In [12]:

```python
#function
def paddle_b_down():
    y = paddle_b.ycor()
    y -= 20
    paddle_b.sety(y)
```

In [13]:

```python
# Keyboard binding
wn.listen()
wn.onkeypress(paddle_a_up, "w")
wn.onkeypress(paddle_a_down, "s")
wn.onkeypress(paddle_b_up, "Up")
wn.onkeypress(paddle_b_down, "Down")
```

In [14]:

```python
# Main game loop
while True:
    wn.update()

    # Move the ball
    ball.setx(ball.xcor() + ball.dx)
    ball.sety(ball.ycor() + ball.dy)

    # Border checking

    # Top and bottom
    if ball.ycor() > 290:
        ball.sety(290)
        ball.dy *= -1

    elif ball.ycor() < -290:
        ball.sety(-290)
        ball.dy *= -1


    # Left and right
    if ball.xcor() > 350:
        score_a += 1
        pen.clear()
        pen.write("Player A: {}  Player B: {}".format(score_a, score_b), align="center", font=("Courier", 24, "normal"))
        ball.goto(0, 0)
        ball.dx *= -1

    elif ball.xcor() < -350:
        score_b += 1
        pen.clear()
        pen.write("Player A: {}  Player B: {}".format(score_a, score_b), align="center", font=("Courier", 24, "normal"))
        ball.goto(0, 0)
        ball.dx *= -1

    # Paddle and ball collisions
    if ball.xcor() < -340 and ball.ycor() < paddle_a.ycor() + 50 and ball.ycor() > paddle_a.ycor() - 50:
        ball.dx *= -1


    elif ball.xcor() > 340 and ball.ycor() < paddle_b.ycor() + 50 and ball.ycor() > paddle_b.ycor() - 50:
        ball.dx *= -1
```

```
---------------------------------------------------------------------------
TclError                                  Traceback (most recent call last)
<ipython-input-14-e661a1992709> in <module>
      4 
      5     # Move the ball
----> 6     ball.setx(ball.xcor() + ball.dx)
      7     ball.sety(ball.ycor() + ball.dy)
      8 

~\Anaconda3\lib\turtle.py in setx(self, x)
   1806         (10.00, 240.00)
   1807         """
-> 1808         self._goto(Vec2D(x, self._position[1]))
   1809 
   1810     def sety(self, y):

~\Anaconda3\lib\turtle.py in _goto(self, end)
   3156                              (self.currentLineItem,
   3157                              self.currentLine[:],
-> 3158                              screen._pointlist(self.currentLineItem),
   3159                              self.items[:])
   3160                              )

~\Anaconda3\lib\turtle.py in _pointlist(self, item)
    753         (9.9999999999999982, 0.0)]
    754         >>> """
--> 755         cl = self.cv.coords(item)
    756         pl = [(cl[i], -cl[i+1]) for i in range(0, len(cl), 2)]
    757         return  pl

<string> in coords(self, *args, **kw)

~\Anaconda3\lib\tkinter\__init__.py in coords(self, *args)
   2467         return [self.tk.getdouble(x) for x in
   2468                            self.tk.splitlist(
-> 2469                    self.tk.call((self._w, 'coords') + args))]
   2470     def _create(self, itemType, args, kw): # Args: (val, val, ..., cnf={})
   2471         """Internal function."""

TclError: invalid command name ".!canvas"
```

In [ ]:

In [ ]: