Python

Noughts and Crosses

All Code Clubs must be registered. Registered clubs appear on the map at codeclub.org.uk - if your club is not on the map then visit jumpto.cc/18CpLPy to find out what to do.

Introduction

Time for another game, and today is Noughts and Crosses, or Tic-Tac-Toe. Players take turns marking an X or an O until one player gets three in a row.

Activity Checklist — Follow these INSTRUCTIONS one by one

Test your Project — Click on the green flag to TEST your code

Save your Project — Make sure to SAVE your work now

Step 1: Drawing the grid

We want to draw four lines, in a # pattern, like this:

```
_|_|_
_|_|_
 | |
```

We could use the turtle commands to draw it, but today we're going to learn about the Tk Canvas.

Activity Checklist

1. Open up IDLE, create a New File, and save it as 'xox.py' ☐

Write the following code:

```
from tkinter import *

main = Tk()

c = Canvas(main, width=600, height=600)
c.pack()

c.create_line(200, 0, 200, 600)
c.create_line(400, 0, 400, 600)

c.create_line(0, 200, 600, 200)
c.create_line(0, 400, 600, 400)

mainloop()
```

1. Save and Run your program, and it will draw a grid ! ☐

The Canvas

We make a 600 by 600 pixel window with `c = Canvas(main, width=600, height=600)`, which looks like this to the computer:

```
   0       200      400      600   ...
  0+--------+--------+--------+-----> across
   |
   |
   |
   |
200|         A         B
   |
   |
   |
   |
400|         C         D
   |
   |
   |
   |
600|
   |
...|
   V
 down
```

Here point `A` is at 200 across, 200 down. Point `B` is at 400 across, 200 down. Point `C` is at 200 across, 400 down. Point `D` is at 400 across, 400 down.

Each of the commands `c.create_line(across1,down1,across2,down2)` draws a line across the screen, and the four numbers act as positions. If we wanted to draw a line from A to D, we would do `c.create_line(200, 200, 400, 400)`.

```
   0       200      400      600   ...
  0 +--------A--------B--------+-----> across
```

```
     |
     |
     |
     |
200  M                          O
     |
     |
     |
     |
400  N                          P
     |
     |
     |
     |
600  |         C         D
     |
...  |
     V
   down
```

We want to draw lines from A to C, B to D, M to O, and N to P.

```
c.create_line(200, 0, 200, 600) # A to C
c.create_line(400, 0, 400, 600) # B to D

c.create_line(0, 200, 600, 200) # M to O
c.create_line(0, 400, 600, 400) # N to P
```

In code, across is often called `x`, and down is often called `y`. This grid system is quite like the co-ordinates you've learned in mathematics, but we start from the top-left hand side.

Step 2: Drawing a O

Activity Checklist

1. In the same file, let's add a new function to draw when you click the mouse!

```
from tkinter import *

main = Tk()

c = Canvas(main, width=600, height=600)
c.pack()

c.create_line(200, 0, 200, 600)
c.create_line(400, 0, 400, 600)

c.create_line(0, 200, 600, 200)
c.create_line(0, 400, 600, 400)

def click(event):
    c.create_oval(200,200,400,400)

c.bind("<Button-1>", click)

mainloop()
```

1. Run your code, and click on the grid, what happens? You should see a circle in the centre of the grid.
2. Let's edit the code, so it will draw where you click. For this we'll need to take the mouse position, and work out which grid square it is in, and change the `click` function

```
from tkinter import *

main = Tk()
```

```
c = Canvas(main, width=600, height=600)
c.pack()

c.create_line(200, 0, 200, 600)
c.create_line(400, 0, 400, 600)

c.create_line(0, 200, 600, 200)
c.create_line(0, 400, 600, 400)

def click(event):
    across = int(c.canvasx(event.x) / 200)
    down   = int(c.canvasy(event.y) / 200)

    c.create_oval(
        across * 200, down * 200,
        (across + 1) * 200, (down + 1) * 200
    )

c.bind("<Button-1>", click)

mainloop()
```

`int(c.canvasx(event.x)/200)` takes the mouse position, `event.x` turns it into the canvas position, `c.canvas(event.x)` and then divides it by 200, and rounds it down, so we get a number 0, 1 or 2 to tell us how far across the mouse is.

1. Run the code, click in the grid squares, each one should fill in with a circle! The code `c.create_oval(across*200,down*200, (across+1)*200,(down+1)*200)` turns 'Along 1, Down 2' into positions on the grid, like Along 200, Down 400. ☐

Step 3: Drawing an X

Activity Checklist

1. In the same file, let's add some code to draw an X, then an O, then an X, ...

```
from tkinter import *

main = Tk()

c = Canvas(main, width=600, height=600)
c.pack()

c.create_line(200, 0, 200, 600)
c.create_line(400, 0, 400, 600)

c.create_line(0, 200, 600, 200)
c.create_line(0, 400, 600, 400)

shape = "O"

def click(event):
    global shape
    across = int(c.canvasx(event.x)/200)
    down = int(c.canvasy(event.y)/200)

    if shape == "O":
        c.create_oval(
            across*200,down*200,
            (across+1)*200,(down+1)*200
        )
        shape = "X"
    else:
        c.create_line(
            across*200, down*200,
            (across+1)*200, (down+1)*200
        )
        c.create_line(
            across*200, (down+1)*200,
            (across+1)*200, down*200
        )
        shape = "O"
```

```
c.bind("<Button-1>", click)

mainloop()
```

1. Run your program, try click on a grid, it should draw a O, click elsewhere it should draw an X We've used a new feature of python, `global` to let us change the variable `shape` in the function `click`. If you change variables defined outside of a function, then you have to use `global`. ☐
2. What happens if you click on the same square twice? This is because our code doesn't keep track of what has been drawn, or where players have moved. We will have write some more code to fix this. ☐

Step 4: Keeping track

Activity Checklist

To stop players playing the same move twice, we'll have to keep track of the moves they make. To do this we'll introduce a list called `grid`

1. In the same file, ☐

```
from tkinter import *

main = Tk()

c = Canvas(main, width=600, height=600)
c.pack()

c.create_line(200, 0, 200, 600)
c.create_line(400, 0, 400, 600)
```

```
c.create_line(0, 200, 600, 200)
c.create_line(0, 400, 600, 400)

shape = "O"
grid = [
    "0", "1", "2",
    "3", "4", "5",
    "6", "7", "8",
]

def click(event):
    global shape, grid
    across = int(c.canvasx(event.x)/200)
    down = int(c.canvasy(event.y)/200)

    square = across + (down*3)

    if grid[square] == "X" or grid[square] == "O":
        return

    if shape == "O":
        c.create_oval(
            cross*200,down*200,
            (across+1)*200,(down+1)*200
        )
        grid[square] = "O"
        shape = "X"
    else:
        c.create_line(
            across*200, down*200,
            (across+1)*200, (down+1)*200
        )
        c.create_line(
            across*200, (down+1)*200,
            (across+1)*200, down*200
        )
        grid[square] = "X"
        shape = "O"

c.bind("<Button-1>", click)
```

```
mainloop()
```

1. Run your program, and try and click in the same square twice? What happens?

Step 5: Finding a winner

Now we have got the game working, we need to find a winner!

Activity Checklist

1. In the same file, we're going to introduce a new function `winner`, and call it to check if the game has been won The completed code looks like this!

```
from tkinter import *

main = Tk()

c = Canvas(main, width=600, height=600)
c.pack()

c.create_line(200, 0, 200, 600)
c.create_line(400, 0, 400, 600)

c.create_line(0, 200, 600, 200)
c.create_line(0, 400, 600, 400)

shape = "O"
grid = [
    "0", "1", "2",
    "3", "4", "5",
    "6", "7", "8",
```

```
]


def click(event):
    global shape, grid
    across = int(c.canvasx(event.x)/200)
    down = int(c.canvasy(event.y)/200)

    square = across + (down*3)

    if grid[square] == "X" or grid[square] == "O":
        return

    if winner():
        return

    if shape == "O":
        c.create_oval(
            across*200,down*200,
            (across+1)*200,(down+1)*200
        )
        grid[square] = "O"
        shape = "X"
    else:
        c.create_line(
            across*200, down*200,
            (across+1)*200, (down+1)*200
        )
        c.create_line(
            across*200, (down+1)*200,
            (across+1)*200, down*200
        )
        grid[square] = "X"
        shape = "O"

def winner():
    for across in range(3):
        row = across*3
        line = grid[row] + grid[row+1] + grid[row+2]
        if line == "XXX" or line == "OOO":
            return True
```

```
    for down in range(3):
        line = grid[down] + grid[down+3] + grid[down+6]
        if line == "XXX" or line == "OOO":
            return True

    line = grid[0]+grid[4]+grid[8]

    if line == "XXX" or line == "OOO":
            return True

    line = grid[2]+grid[4]+grid[6]

    if line == "XXX" or line == "OOO":
            return True


c.bind("<Button-1>", click)

mainloop()
```

1. Try playing the game and winning, can you make any more moves? We have four checks in `winner`

   - Check each row for three X's or O's
   - Check each column for three X's or O's
   - Check the diagonal from left to right
   - Check the diagonal from right to left.

Try

You're all done! Why not change the code to draw different shapes!