Level 3

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

Introduction

Today we're going to try and write some code so that the computer can play noughts and crosses. It won't play well at first, but soon it might even win against you!

Step 1: Start with what we have from last week

In Lesson 6, we wrote a two player noughts and crosses game, using the "Tk Canvas" to draw on the screen, before we start writing new code, we're going to need the basic part of the game.

Activity Checklist

1. Open IDLE, and either open your file from last week and save it as a new file, or if you can't find it, copy the following in: ☐

```
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)
```


These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

```
    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
```


These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

```
    line = grid[2]+grid[4]+grid[6]
    if line == "XXX" or line == "OOO":
            return True
c.bind("<Button-1>", click)
mainloop()
```

2. Save and run the program, make sure it works! ☐

You should be able to click on squares, to place circles and crosses until there are three in a row.

Step 2: Play anywhere

Before we can teach the computer how to make a good move, we should teach it to make any move. We will get the computer to pick any free square and play there.

Remember that we use the variable grid to store how the board looks. It's a list that starts off `["0", "1", "2", ... ]`, and we replace elements with "X" or "O" as the game progresses. We will have to find a free spot, then play it.

Activity Checklist

1. At the top of the file, we will have to import the random library, so we can pick a move: ☐

```
from tkinter import *
import random
```

You hopefully remember using `random.choice` from the earlier lessons.

In the same file we will introduce a new function, `free_squares`:

2. Add this code below the function `winner`, and above the line `c.bind(...)`: ☐

```
def free_squares():
    output = []
    for position, square in enumerate(grid):
        if square != "X" and square != "O":
            output.append(position)
    return output
```

This function creates an empty list, and then for each square in the grid, checks if it is empty.

These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

We use a function `enumerate` to tell us the positions of each item in the list. enumerate turns a list of ['A','B','C'] into pairs of (0, 'A'), (1,'B') and so on, so we don't have to count them.

Next we write a function `play_move()` which finds an empty square to play in.

3. Add this function after `free_squares` and before the line `c.bind(...)` ☐

```
def play_move()
    moves = free_squares()
    move = random.choice(moves)

    down = move // 3
    across = move % 3

    grid[move] = "X"
    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
    )
```

First we get the list of empty squares, pick one, and convert the square number into across and down, using the `%` and `//` operators. Let's look at the numbered grid to see how this works:

```
     0 1 2
     -----
0  | 0 1 2
1  | 3 4 5
2  | 6 7 8
```

The 5 square is 1 down, and 2 across. If we divide 5 by 3, we get 1 with remainder 2

`5 // 3` is 1, `6 // 3` is 2, and so on. The `//` operator gives us how many times 3 divides it, but ignores the remainder, which tells us how far down we must go.

5 `%` 3 is 2, 6 `%` 3 is 0. The `%` operator gives us the remainder, which is how far along we must go.

4. Now we will edit the `click()` function to use `play_move`, so after you click, the computer makes a move. ☐

These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Keep track of your progress by ticking off the boxes below:

```
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

    c.create_oval(
        across*200,down*200,
        (across+1)*200,(down+1)*200
    )
    grid[square] = "O"

    if winner():
        return

    play_move()
```

We check to see if the player has won, and if not, the computer will play a move!.

5. Save your program and run it. ☐

The computer should play a move after you do. It should not play the game very well.

Step 3: Pick a move that wins

The computer can play noughts and crosses, but badly. Let's help it a little. Instead of picking a random move, let's make it pick a move that wins, if it sees one. The idea is that we can try each move in turn and see if it wins, and then play it.

Activity Checklist

1. Edit the winner function to take an argument grid: ☐

```
def winner(grid):
    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
```


These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

```
    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
```

You should only have to change the first line of the function. This means winner will use a grid passed to it, instead of the grid of the current game

2. Now we change click to pass in this grid. ☐

```
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(grid):
        return
    c.create_oval(
        across*200,down*200,
        (across+1)*200,(down+1)*200
    )
    grid[square] = "O"
    if winner(grid):
        return
    play_move()
```

3. Every time you see `winner()`, you replace it with `winner(grid)`. ☐
4. Run your code, it should work like before. It is important to make sure we haven't made any mistakes. ☐
5. Let's change `play_move` to find a winning move! ☐

These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

```
def play_move():
    move = -1
    moves = free_squares()
    if not moves:
        return

    # find winning move if exists
    for possible in moves:
        new_grid = list(grid)
        new_grid[possible] = "X"
        if winner(new_grid):
            move = possible
            break

    if move <0:
        move = random.choice(moves)

    across, down = move%3, move//3

    grid[move] = "X"
    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
    )
```

We make a copy of the grid, using `list(grid)`, place an **X** where we could play, and call winner to see if it wins!

6. Run and test your program. If the computer is lucky, it should try and win. ☐

Things to Try

Try playing a few games and seeing what happens.

Step 4: Pick the move that blocks

The other strategy we will use, is to look for a winning move for the player, and play it instead. i.e. to block any potential three in a row.

These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Level 3

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

Activity Checklist

1. Edit `play_move` to find the players winning move, and block it!

```
def play_move():
    move = -1
    moves = free_squares()
    if not moves:
        return

    # find winning move if exists
    for possible in moves:
        new_grid = list(grid)
        new_grid[possible] = "X"
        if winner(new_grid):
            move = possible
            break

    if move < 0:
        for possible in moves:
            new_grid = list(grid)
            new_grid[possible] = "O"
            if winner(new_grid):
                move = possible
                break

    if move <0:
        move = random.choice(moves)

    across, down = move%3, move//3

    grid[move] = "X"
    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
    )
```

2. Run your code, and try to win.

It should be a lot harder to beat the computer.

These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Keep track of your progress by ticking off the boxes below:

The Complete Program

Your final program should look something like this!

```
from tkinter import *
import random
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)

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(grid):
        return

    c.create_oval(
        across*200,down*200,
        (across+1)*200,(down+1)*200
    )
    grid[square] = "O"

    if winner(grid):
        return

def winner(grid):
```


These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

```
    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

def play_move():
    move = -1
    moves = free_squares()
    if not moves:
        return

    # find winning move if exists
    for possible in moves:
        new_grid = list(grid)
        new_grid[possible] = "X"
        if winner(new_grid):
            move = possible
            break

    if move < 0:
        for possible in moves:
            new_grid = list(grid)
            new_grid[possible] = "O"
            if winner(new_grid):
                move = possible
                break

    if move <0:
        move = random.choice(moves)

    across, down = move%3, move//3
```


These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!

Playing Against the Computer

Keep track of your progress by ticking off the boxes below:

```
    grid[move] = "X"
    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
    )
def free_squares():
    output = []
    for position, square in enumerate(grid):
        if square != "X" and square != "O":
            output.append(position)
    return output
c.bind("<Button-1>", click)
mainloop()
```

Challenge

It's still possible to win against the program, but can you make changes to it to make it play perfectly?

These projects are for use inside the UK only. All Code Clubs must be registered. You can check registered clubs on the map at **www.codeclub.org.uk**.
This coursework is developed in the open on GitHub, at **www.github.com/CodeClub/**. Come and join us!