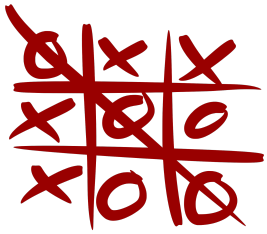
Let's Code a Game: Tic Tac Toe!

Today we're going to be making a text based version of the game Tic Tac Toe, also known as "Naughts and Crosses".



Part 0: Getting set up!

To get started first we need to create a file where we are going to write the code for our game.

1) In your Python 3 IDLE click ***File*** and create a ***New File***.
2) Start by saving your file and calling it ***tictactoe.py***

Now when you want to run your code, just click ***Run*** (or press ***F5*** on the keyboard)!

Great! Now we're ready to code!

Part 1: Printing the grid

In order to play Tic Tac Toe, we are going to need to print out the game board.

We want to print a grid that looks like this:

```
-------------
| 1 | 2 | 3 |
-------------
| 4 | 5 | 6 |
-------------
| 7 | 8 | 9 |
-------------
```

Each square has a number in it so we can tell the computer which square we want!

To do this, we'll need to use `print`. Want a hint on how to print? Look at your cheat sheet!

Task 1.1

In your file ***tictactoe.py*** write some code that will `print` out the grid above.

Hint: You might need to use multiple `print` *statements*
Hint: There are 13 dashes in the first line,
Hint: What happens when you multiply a character by a number?

Part 2: Storing the players' names

We need to remember the name of the two players. Let's store each name in a variable.

Use `input` to ask each player for their name and store them in variables.
(add it above your printer).

Task 2.1

Before you print the grid, ask these 2 questions using `input` to get the names of the players:

- `"Who will be naughts? "`
- `"Who will be crosses? "`

Name your variables `player0` for naughts and `playerX` for crosses.

Now you have the names, you'll need to show them!

Task 2.2

Print a message welcoming both of the players by their name. Here's an example of how your code should run:

```
Who will be naughts? Julia
Who will be crosses? Maddie
Welcome Julia and Maddie!
```

Remember how we learnt to add strings together?
We'll need to do that here to add all the names and text together!

Part 3: Storing the game

We're going to store our game board in lists!

When we start the game we have no naughts or crosses.
Remember we started out by printing a grid with the numbers 1-9 in it.

```
-------------
| 1 | 2 | 3 |
-------------
| 4 | 5 | 6 |
-------------
| 7 | 8 | 9 |
-------------
```

Let's store these numbers in our list to start with. Later they will be replaced with X's and O's.

Task 3.1
Make the list to store the naughts and crosses. Let's call it the `grid`.

Fill `grid` with the numbers 1 to 9, each of the numbers should be stored as a string.

E.g. `grid = ["1", "2", "3" ... "9"]`

Part 4: Make a smarter printer!
Our existing printer is great for printing out the grid at the start of the game, but it prints the same grid every time! No fun for a game that changes!!

We've already got `grid` to store our game board in.
```
grid = ["1", "2", "3", "4", "5", "6", "7", "8", "9"]
```

Now we just have to figure out to get each item out of `grid` and into the right spot on the grid!

Task 4.1
Change your existing print code to get the value for each square from `grid`.
Remember to use:
- adding strings together
- looking up specific items in a list (remember to count from 0!)

If your new printer works the printout will look exactly the same as before!!

Test: You can test that your new changeable printer works by putting something different (like a X or O) in `grid` and seeing if it prints out something different than before!

Part 5: The first turn of the game!

Now we are getting to the exciting part! We get to start making the game playing part!
To put a piece down, we need to know two things:

- Whether it is a O or an X
- Where the O or X goes

Asking for the symbol

We need to find out what symbol to use when a player takes a square.
When you play naughts and crosses you will be using 'X' and 'O', but really, you can put in anything.

Task 5.1

1. Ask the player what symbol to place using `input`.
2. Store it in a variable named `input_symbol`.

Ask which square they want to take

The next thing we need to do is find out which square the player wants to put their symbol in.
We can do this using `input`.

Task 5.2

1. Show the players which squares are still available by printing out the `grid`. Any squares with numbers in them are free and can be selected.

2. Then ask them which square they want to take using `input`. Store it in a variable named `input_square`.

Turning that input into a useful number!

We now know which square the player wants to take. It's stored in `input_square`; the only problem is it's not a number, **it's a string!!**

We can't do maths or counting using a symbol (even though it is a number symbol)! We need to convert it to a number first.

Task 5.4

Transform `input_square` to a number and store it in a new variable `input_square_int`
Hint: Do you remember how we used `int()` to turn a string into a number?

Updating grid

Before we can update `grid` with the symbol to play we need to work out which item in the list we want to update. **This is a bit tricky because we need to start counting from 0.**

```
-------------                        Count from 0 !
| 1 | 2 | 3 |       0    1    2    3    4    5    6    7    8
-------------
| 4 | 5 | 6 |    ["1", "2", "3", "4", "5", "6", "7", "8", "9"]
-------------
| 7 | 8 | 9 |
-------------
```

We can see that the indexes don't quite line up with the numbers in our grid because we have to start counting from 0 when using a list.

When we want to put a X in square 1, that is actually item 0 in the list.
If we want to put a X in square 9 that is actually item 8 in the list.

Putting an X in square 9:

```
-------------                        Count from 0 !
| 1 | 2 | 3 |       0    1    2    3    4    5    6    7    8
-------------
| 4 | 5 | 6 |    ["1", "2", "3", "4", "5", "6", "7", "8", "X"]
-------------
| 7 | 8 | X |
-------------
```

For the example above, `input_square_int` is 9 but we want `grid_index` to be 8. We just need to write some code to do that for us. *(P.S. "Index" is the name we give to a position in a list!)*

Task 5.5

We need to find the index that corresponds to the value in `input_square_int` and store it in `grid_index`. So if the user wants to edit square 5, `grid_index` should be 4.

Which of these options do you think will work?

```
a. grid_index = input_square_int - 1
b. grid_index = input_square_int - 2
c. grid_index = input_square_int + 1
d. grid_index = input_square_int + 2
```

Add it to your code after you get `input_square_int`

Now we know which list item we need to update! Let's get updating!

Goal: Update the the chosen index of grid to the chosen symbol

Task 5.6
Add in a line of code to your game that updates the `grid` at the right location with a new symbol.

You'll need to use:
- `grid`
- `grid_index`
- Updating a specific index of a list
- `input_symbol`

Print the grid again!

Goal: Print out the updated game board!

Task 5.7
Copy and paste the same code that you used to print the board in **Part 1** to print your new updated grid!

Bonuses:
1. See what happens if you play a turn and put in something that's not an X or O. Try putting the Letter A in or something!

2. Trying putting in something like :) or your name instead of a symbol! What happens?

3. Can you find any other ways to break it?

Don't worry as we add more code there'll be less ways to break it!

Part 6: Now repeat!

Now we have finished one turn of the game. The rest of the game is basically just doing the same thing over and over again!

But writing out all of that code again would be boring and computer programmers are known for being lazy! So instead we are going to use a `while` loop to make our code run for more than one turn!

Goal: Use a while loop to make the game continue for 9 turns.

Task 6.1
Above the code you wrote for **Part 5**, set up a variable to keep track of how many turns it has been and call it `turns`. At the start of the game there has been 0 turns.

Task 6.2
1. Create the `while` loop. Tell the while loop to run until there has been 9 turns.
2. Indent your code from **Part5** and **Part6** to be inside your while loop.

Task 6.3
Inside the loop you just created, at the end of the turn, update the number of turns it's been.
Hint use maths!

Bonuses:
1. Try playing a game, but chose a square that's already taken!

2. Try playing only X's and no O's

3. What happens if you play only on the same spot? When does the game finish?

When you write a program you can rig it however you want! To make it unriggable we need more code!

Are you finished with all of this?

Let a tutor know. We've got more stuff for you!