Lesson 14: Drawing a Smiling Minion

In this lesson, we'll learn how to draw a character that will smile when we click the screen!

PART 1: Setting up Tkinter

Like before, we'll first import the `tkinter` and `set up the canvas`.

Try typing the code below into a new program:

```
from tkinter import *
myTk = Tk()
smiling = False
canvas = Canvas(myTk, width=400, height=400)
canvas.pack()
```

This time, we've also made a variable called `smiling` that will be set to whether or not the character should be smiling currently.

PART 2: Making the Main Loop

Next we'll make the main loop of our program, where we'll delete and draw everything.

Try adding the following code to your program:

```
canvas.pack()
while True:
    canvas.delete("all")
    myTk.update()
myTk.destroy()
```

Once we've made our drawing functions, we'll add more to this loop.

PART 3: Drawing the Character

Now, we'll create a function that draws our character! In our function, we'll draw a lot of different shapes on top of each other that will make up a picture of the character.

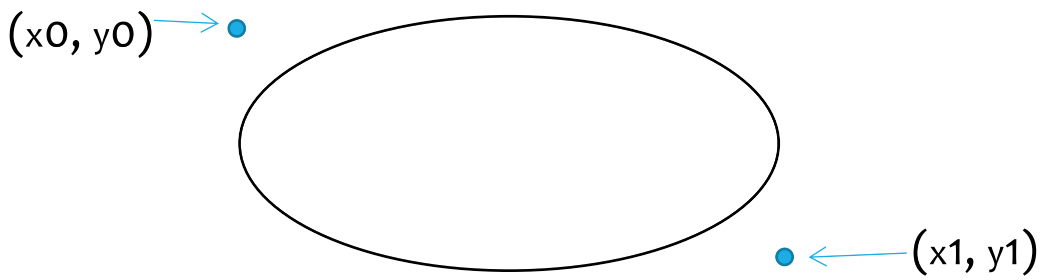
First, we'll draw a yellow oval and rectangle for the base shape, and then we'll add more shapes for the goggles and eye. Drawing functions called lower in the code will be on top.

Since we want our character's mouth to move, we won't draw the mouth just yet though.

Creating an oval: When we create an oval, we will use this command:

```
canvas.create_oval(x0, y0, x1, y1)
```

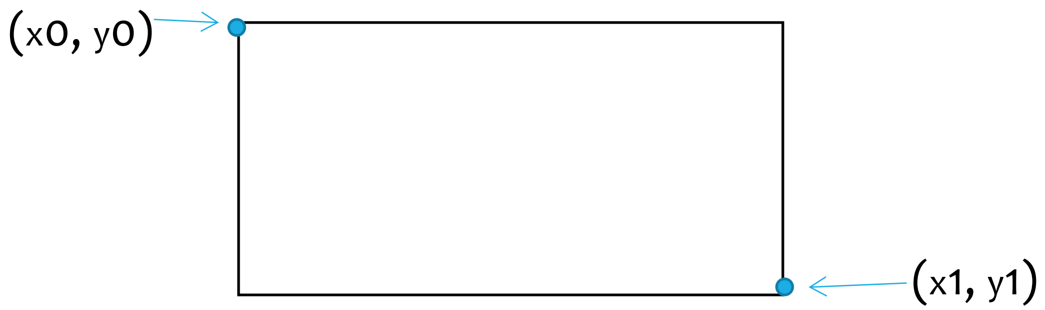
The coordinates are shown below:



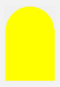
Creating a rectangle: When we create a rectangle, we will use this command:

```
canvas.create_rectangle(x0, y0, x1, y1)
```

Each rectangle is specified as two points: (x0, y0) is the top left corner, and (x1, y1) is the location of the pixel on the bottom right corner as shown below:



Try adding the following code to your program:



```
canvas = Canvas(myTk, width=400, height=400)
canvas.pack()
def drawcharacter():
    canvas.create_oval(150, 150, 250, 250, fill="yellow", width=0)
    canvas.create_rectangle(150, 200, 250, 300, fill="yellow",
                                  outline="yellow")
    canvas.create_rectangle(150, 195, 250, 205, fill="black")
    canvas.create_oval(175, 175, 225, 225, fill="gray", width=0)
    canvas.create_oval(180, 180, 220, 220, fill="white", width=0)
    canvas.create_oval(195, 195, 205, 205, fill="black", width=0)
while True:
```

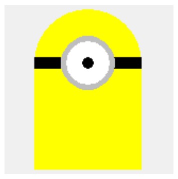
The `width` parameter sets the width of the shape's outline. (`width=0` makes it invisible.)

Before we can see our character, we'll need to call this function in the main loop.

Add this line to your main loop:

```
while True:
    canvas.delete("all")   # Deletes all of the shapes
    drawcharacter()        # Draws the character's shapes back
    myTk.update()
```

If you run your program now, your character should look like this:



PART 4: Drawing a Mouth

Next, we'll add a function that draws a mouth on our character!

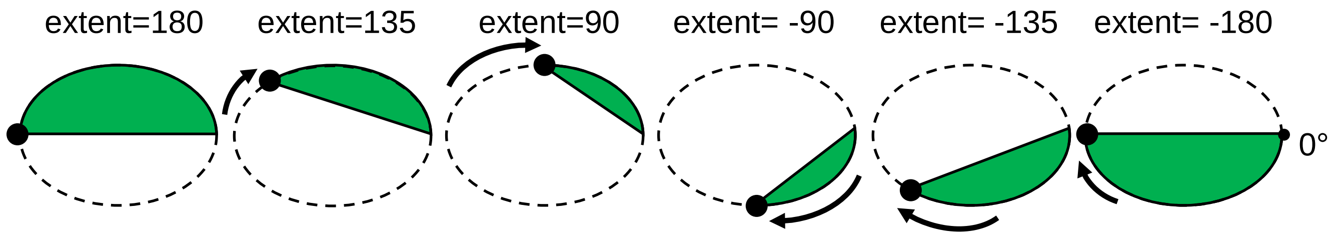
In order to make the character smile when the mouse is clicked, we'll want to program two different ways of drawing the mouth.

Try adding the following code above the main loop:

```
def drawcharacter():
    . . .
    canvas.create_oval(195, 195, 205, 205, fill="black", width=0)
def drawmouth(smile):
    if smile:
        canvas.create_arc(175, 230, 225, 260, extent=-180,
                              style=CHORD, fill="black")
    else:
        canvas.create_rectangle(175, 250, 225, 255, fill="black")
while True:
```

Our function requires a parameter for whether or not the character should be smiling.

If they should be smiling, we'll use the `create_arc()` function to draw the smile.
If they shouldn't be smiling, we'll just draw a black rectangle for a mouth.



`create_arc()` makes an arc (a section cut from an oval) based on the style and extent.
Using the `CHORD` style, `create_arc()` will draw the portion of the oval from `0°` to `extent`.

Again, before we can see the mouth, we'll need to add more code to the main loop!

Try adding this line to the main loop:

```
while True:
    canvas.delete("all")
    drawcharacter()
    drawmouth(smiling)
    myTk.update()
```

Here, we're passing the variable `smiling`, that we defined near the top of the program.

If you test your program now, your character should also have a mouth!

PART 5: Detecting Mouse Clicks

So far, our character has a mouth, but they won't be able to smile just yet, because `smiling` has only been set to `False`.

To change this, we have to set `smiling` to `True` whenever the mouse clicks the screen.

The `Tkinter` module lets us detect mouse clicks by using an *event handler* system.
An event handler is a system that waits for *events* (like mouse clicks or key presses), and calls certain functions when those events happen while running the program.

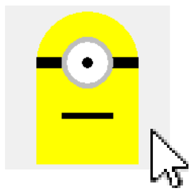
We can tell `Tkinter` what functions to call by *binding* a function to an event.
Binding an event is how we tell a program what function to call when an event happens.

Try adding the following code above your main loop:

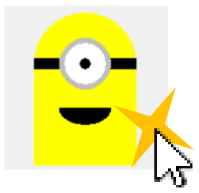
```
def drawmouth(smile):
    .  .  .
        canvas.create_rectangle(175, 250, 225, 255, fill="black")
myTk.bind("<Button-1>", mouseclick)
myTk.bind("<ButtonRelease-1>", mouserelease)
while True:
    .  .  .
```

Here, we've called the `bind()` function on `myTk` twice, to bind the mouse press and mouse release events to different functions.

In this case, we've bound the `<Button-1>` (mouse click) event to a function called `mouseclick()` and the `<ButtonRelease-1>` (mouse release) event to `mouserelease()`.



Mouse is released → calls `mouserelease()`



Mouse pressed → calls `mouseclick()`

Next, we'll need to actually define the functions `mouseclick()` and `mouserelease()`.

Try adding the following code near the top of your program:

```
canvas.pack()
def mouseclick(event):
    global smiling
    smiling = True
def mouserelease(event):
    global smiling
    smiling = False
def drawcharacter():
    . . .
```

Here, we've defined the functions `mouseclick()` and `mouserelease()` that we bound.

Because of how the event handler system in `Tkinter` works, these functions need to have a parameter for the event being called, which is passed as an `event` object.

The `event` object will automatically be passed to our functions by the event handler whenever the `<Button-1>` or `<ButtonRelease-1>` events happen.

The `global` keyword tells our program to use a variable we defined outside of this function, instead of defining a new variable. This lets us modify the value of a variable that was defined outside of the function.

Inside each of our functions, we're simply changing the value of the variable `smiling` to either `True` or `False`. This way, when our main loop calls `drawmouth(smiling)`, it will tell `drawmouth()` whether to draw a smile or not.

Now, when you run your program, you should be able to make your character smile whenever you click the screen!

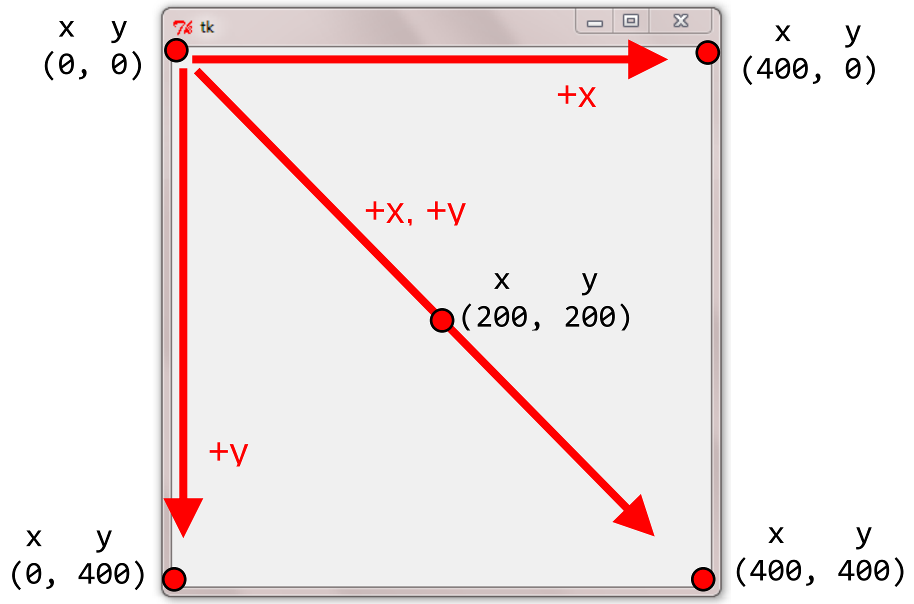
PART 6: BONUS: Making your own Character

Using what you've learned in this lesson, you can make your own characters too!

All of the `Tkinter` drawing functions follow this format for their parameters:

```
canvas.create_shape(150, 200, 250, 300, fill="yellow",  . . .)
                    x1   y1   x2   y2         color       others
```

Where `(x1, y1)` is one of the corner's coordinates, and `(x2, y2)` is the other corner's.



To make a different character, make sure to **go to File > Save As, and save your program as a new file**.

After this, try replacing the lines in `drawcharacter()` to make a new character!

Here are some examples of characters you could try drawing:

Dog, Cat, Pig, Elephant, Tiger, Fish, Smiley Face

Once you've drawn your character's base, you can add code to make them blink, smile, raise eyebrows, wave an arm or wag their tail when you click the screen!