Advanced Python
Lesson 6: Drawing an Alien using Tkinter

In this lesson, we will be learning about Tkinter and keyboard events to draw and control our very own alien. By the end, we should have an alien with some functionality.

PART 1: Setting up Tkinter

In order to use any of the functions in the `Tkinter` module, we'll need to import it first. Then we will need to make a window for our drawing to go so we use the tkinter function Tk() which makes a new tkinter window. We then name the window and create a canvas which will be where we create our shapes for our drawing.

The function `window.mainloop()` must **ALWAYS** be the last line. It will update the tkinter window with any changes or drawings.

Try typing the code below into a new program:

```
from tkinter import *
window = Tk()
window.title("Alien")
c = Canvas(window, height=300, width = 400)
c.pack()
window.mainloop()
```

Once you type that out, save and run your code. A window should pop up with your title.

PART 2: Drawing Shapes

The Alien drawing we will make will be a collection of shapes. In order to draw shapes to our tkinter screen, we will need to use functions inside our canvas variable c.

The shapes are layered based on their order of execution. A square made on line 1 will be under a circle made in line 2. Shapes also need a fill color to be selected.

There are many different drawing functions, here are some of them:

```
canvas.create_line(x1, y1, x2, y2)

canvas.create_oval(x1, y1, x2, y2, fill = “color”)

canvas.create_polygon(x1, y1, x2, y2, x3, y3, fill = “color”)

canvas.create_text(x1, y1, text = “Message”)
```

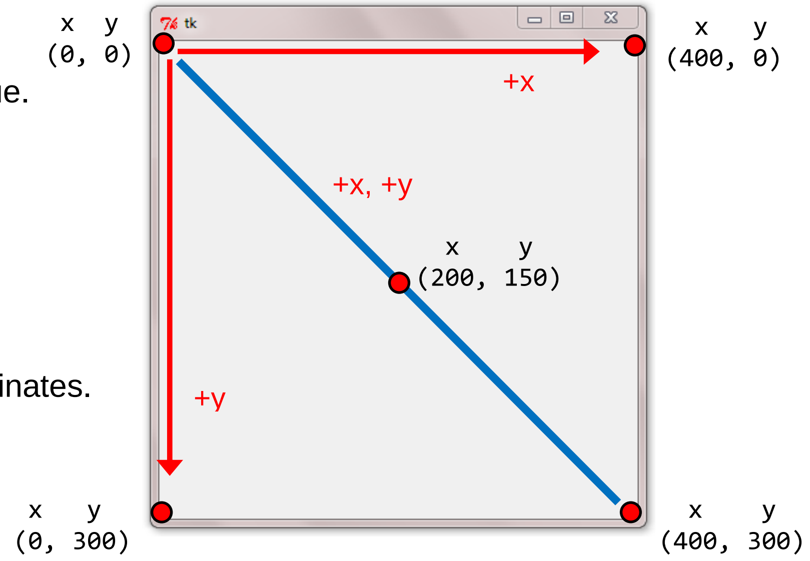
We made the canvas a size of:
Width = 400 and Height = 300

A **point** on the canvas has a (x,y) value.
We call this the coordinate.

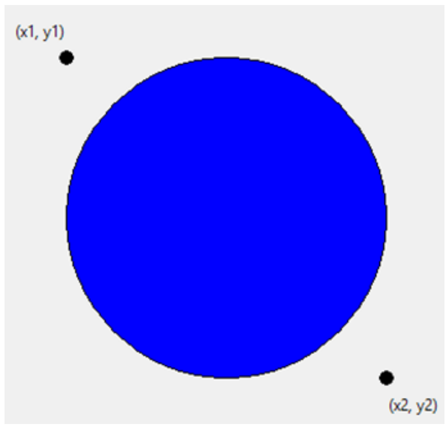
A **line** requires two coordinates.
The line connects the coordinates.

A **text** requires a center coordinate.

A **polygon** will connect multiple coordinates.
This is useful for drawing triangles.



An **oval** is different because it requires two coordinates.
The coordinates create a box for the circle to be enclosed in.
The first coordinate is the top left point of the box.
The second coordinate is the bottom right point of the box.



These functions will return a number that represents which shape it is in the canvas.

We can use this number later to change the color of a certain shape, so we will save each shape in a variable name associated to it.

Try typing the code below into your program:

```
body = c.create_oval(100, 150, 300, 250, fill = "green")

eye = c.create_oval(170, 70, 230, 130, fill = "white")

eyeball = c.create_oval(190, 90, 210, 110, fill = "black")

mouth = c.create_oval(150, 220, 250, 240, fill = "red")

lips = c.create_line(150, 230, 250, 230)

eyeneck = c.create_line(200, 150, 200, 130)

hat = c.create_polygon(180, 75, 220, 75, 200, 20, fill = "blue")

words = c.create_text(200, 280, text = "I am an alien!")

window.mainloop()
```

Save and run your code. You should see an alien on the tkinter screen.

PART 3: Functions

We will now create some functions to make some changes to our alien.

A **function** is a block of code that we can call. There are a lot of built in functions that we have been using already. The most popular is the print() function.

In order to make our own function, we need this syntax to define our function:

```
def my_function():
     #stuff my function will do
```

And in order to use the function, we call the function with this code:

```
my_function()
```

Another function we will need to learn about to change a canvas shape information is the canvas function **itemconfig().** This needs a shape reference number which we stored when we first created our shapes, and also needs the specific info we are changing.

For example, to change the hat color to green we use this code:

```
c.itemconfig(hat, fill = “green”)
```

Try typing the code below into your program:

```
def mouth_open():
    c.itemconfig(mouth, fill = "black")
def mouth_closed():
    c.itemconfig(mouth, fill = "red")
def blink():
    c.itemconfig(eye, fill = "green")
    c.itemconfig(eyeball, state = HIDDEN)
def unblink():
    c.itemconfig(eye, fill = "white")
    c.itemconfig(eyeball, state = NORMAL)
window.mainloop()
```

Notice we change the eyeball state variable to HIDDEN which will make the circle invisible. We can make the eyeball reappear by changing the state to NORMAL.

You can test these functions by calling them before the last line. You can also make some more functions if you want, like make his hat go away and make him say “Give my hat back!”

PART 4: Event Functions

We now want some functions to be called when a certain event happens like a mouse click. In order to bind that event to a function, we need to make event functions. The only difference from regular functions is that event functions need a variable passed in.
To do this we add an event variable in between the parentheses.

Try typing the code below into your program:

```
def burp(event):    #We need that event variable to bind
    mouth_open()
    c.itemconfig(words, text = "BURP!")
window.mainloop()
```

PART 5: Mouse Binding

Now that we have our burp event function, we are ready to bind it to an event. We will bind the left mouse click to our burp function.

In order to bind an event to a function, we need to use a canvas function called bind_all().

The syntax for this function looks like this:

`canvas.bind_all("<Event_Name>", event_function_name)`

Try typing the code below into your program:

```
c.bind_all("<Button-1>", burp)
window.mainloop()
```

Save and run your code. If you click the left mouse button, the alien should burp. If you want, try making an event function for unburp() which will close your mouth and say "Excuse me!"

PART 6: Key Binding

Now that you know how to bind an event, you can bind all other type of events. The tricky part is knowing the event name. We will now add event function to make our alien fall asleep and wake up. We will also bind these functions to key events for the "a" and "z" key.

Try typing the code below into your program:

```
def asleep(event):
    blink()
    c.itemconfig(words, text = "ZZZZZZ")
def awake(event):
    unblink()
    c.itemconfig(words, text = "Good Morning")
c.bind_all("<KeyPress-a>", asleep)
c.bind_all("<KeyPress-z>", awake)
window.mainloop()
```

Save and run your code. The "a" and "z" key should now animate the alien's eye.

PART 7: Bind Function to all Keys

We will now try something a little different where we bind an event function to all key events. Inside the event function we will determine which key is pressed. If an arrow key is pressed, we will move the eyeball in that direction.

To do this we will need to use another canvas function **move()** which looks like this:

```
canvas.move(shape_number, x, y)
```

Where x and y is the amount to move the shape's coordinates.

We will also now be using the event variable we pass into our function. The event variable has information about the event like where the mouse is and which key is pressed. We will use the event's **keysym** variable to determine what key is pressed.

Try typing the code below into your program:

```
def eye_control(event):
    key = event.keysym
    if key == "Up":
        c.move(eyeball, 0, -1)
    elif key == "Down":
        c.move(eyeball, 0, 1)
    elif key == "Left":
        c.move(eyeball, -1, 0)
    elif key == "Right":
        c.move(eyeball, 1, 0)
c.bind_all("<Key>", eye_control)
window.mainloop()
```

Save and run your code. You should now be able to move the eyeball of alien with the arrow keys. Think of some other functions and events you want and try them on your own alien.