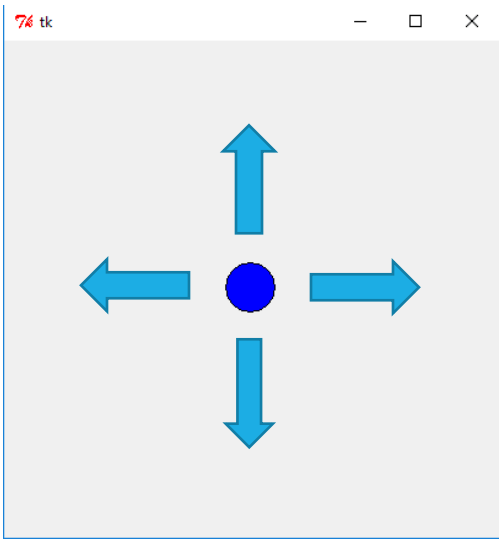
Lesson 15: Moving Objects in Tkinter

In this lesson we'll create a program that lets us move an object around on the screen.

After we finish making a moveable object, we'll use what we've learned to make a game!



PART 1: Setting up Tkinter

Type the code below into a new program and fill in the blanks:

```
from tkinter import *
import time    #we need this module for our loop to wait
myTk = Tk()
canvas = Canvas(myTk, width=400, height=400)
canvas.pack()
moveleft = False          #if true we move left
_________________         #if true we move right
_________________         #if true we move up
_________________         #if true we move down
```

We'll use the `time` module again to make sure the main loop doesn't run too quickly.

After packing the canvas, we'll define four variables to represent whether or not the player should be moving left, right, up, or down. (`True` means they'll move that way)

PART 2: Adding the Main Loop

Next, we'll add the main loop for our program, which will look similar to earlier main loops.

Look at the smiling minion lesson, if you can't remember how to update myTk.

Try adding the following code at the end of your program and fill in blank:

```
movedown = False

while True:

    _____________    #we need to update myTk for the movement

    time.sleep(0.01)    # Wait 1/100th of a second before the next repeat

myTk.destroy()
```

The code `time.sleep (0.01)` will force our program to wait a 1/100th of a second before updating the object movement. Without this code, our object would most likely move the speed of light because it's moving thousands of times per seconds.

`time.sleep()` is very useful when you want your program to stop a few seconds, or have a short delay in the loop.

PART 3: Creating a Shape to Move

Next, we'll create a circle that we want to move around the screen.

Try adding this line above your main loop:

```
movedown = False

c = canvas.create_oval(180, 180, 220, 220, fill="blue")

while True:
```

This code will create a blue circle centered at (200,200) with a radius of 20 pixels and then store the object into the variable `c`.

When we call `canvas.create_oval()`, the function actually returns a value.
This value represents whatever canvas element was just created.

In order to move that element (the oval) around, we'll need to store the returned value in a variable, which we've named `c`.

PART 4: Moving our Shape

Now, we'll use the variable `c` with the `canvas.move()` function to move our circle!

We will be creating our own function called movement() that will be in charge of moving the circle in a direction if the variables we created earlier are True. (ex. moveleft)

We need to input three variables for `canvas.move()` to move our circle.

- The object variable that needs to be moved (our variable `c`)
- The pixel movement in the x axis (negative goes left, positive goes right)
- The pixel movement in the y axis (negative goes up, positive goes down)

Try adding the following code above your main loop and fill in the blank:

```
c = canvas.create_oval(180, 180, 220, 220, fill="blue")
def movement():
    if(moveleft):    #if we need to move left
        canvas.move(c, -3, 0)  #move c 3 pixels to the left
    if(_________):  #if we need to move right
        canvas.move(_,__,__)   #move c 3 pixels to the right
    if(______):     #if we need to move up
        canvas.move(_,__,__)   #move c 3 pixels up
    if(________):   #if we need to move down
        canvas.move(_,__,__)   #move c 3 pixels down
while True:
```

After this, we'll modify our main loop to call the movement function every update.

Add this line to your main loop:

```
while True:
    movement()
    myTk.update()
```

Now our circle should move, if we set any of our direction variables to True. **Assign one of the direction variables to True, and test your program**. It should fly off the screen.

Make sure all the direction values are set to False after you test your code!

PART 5: Binding Keyboard Events to Movements

Just like we bound a mouse click event to make our character smile in the last lesson, we can bind keyboard events to call functions in Python as well!

We will need to check if the arrow keys are pressed or released in order to correctly change the direction variables to True or False.

First, we'll create a function that will get called whenever a key is pressed.

Try adding the following code to your program:

```
c = canvas.create_oval(180, 180, 220, 220, fill="blue")
def keypress(event): #function to activate movement from keys
    global moveleft, moveright, moveup, movedown
    if(event.keysym == "Left"):
        moveleft = True
    if(event.keysym == "Right"):  # Copy and paste these two lines
        moveright = True          # twice more for Up/ Down
def movement():
```

When we want to set multiple variables from outside a function, we can use the `global` keyword and separate the variables with commas.

The `event` parameter will be passed to our keypress function whenever a key is pressed.

The `event` object contains the information about what key is pressed, where the mouse is on the screen, and other information.

The information we are looking for is what key is pressed which is stored in a variable `keysym`, which stands for *key symbol*.

A key symbol is the name of whatever key was pressed that triggered the event.

The arrow keys have key symbols that relate to their direction:

("Up", "Right", "Down", "Left")

We can use the key symbol to check what key triggered the key press event, and if that key was an arrow key, then we'll set the variable for that direction to `True`.

Next, we'll make a similar function to set the variables to `False` when a key is released.

Try adding the following code below the previous function:

```
def keypress(event):
    . . .
    if(event.keysym == "Down"):
        movedown = True
def keyrelease(event): #function to deactivate movement from keys
    global moveleft, moveright, moveup, movedown
    if(event.keysym == "Left"):   # Copy and paste these two lines
        moveleft = False          # three times for Right/Up/Down
def movement():
```

PART 6: Binding the Keyboard Events

The last step is to actually bind our keyboard events so our program will respond to them.

Add these lines above your program's main loop:

```
            canvas.move(c, 0, 3)
canvas.bind_all("<KeyPress>", keypress)
canvas.bind_all("<KeyRelease>", keyrelease)
while True:
```

The `bind_all()` function binds a function to the whole application, which is necessary for keyboard events (`bind()` won't work).

Pressing any key will now activate the keypress function, and releasing a key will activate the keyrelease function.

Now, your program should let you move your circle around the screen!