Lesson 16: Pokemon Safari Game

In this lesson, we will be creating a Pokemon Safari game using Tkinter and our previous lesson with the moving circle.

Our game will be like Pokemon Go. Our blue circle has to travel to different parts of the screen to find and catch all the Pokemon before time runs out.

PART 1: Saving our Moving Circle as New Project

We will be using all the code we used in our last lesson of moving the circle with the arrow keys. Open that code back up, and then press File->Save As and type in a new name for this project "pokemon_safari.py"

Now you should still have the code for the moving ball and a duplicate of that code with a different file name.

Run your code to make sure you can still move the circle with the arrow keys correctly.

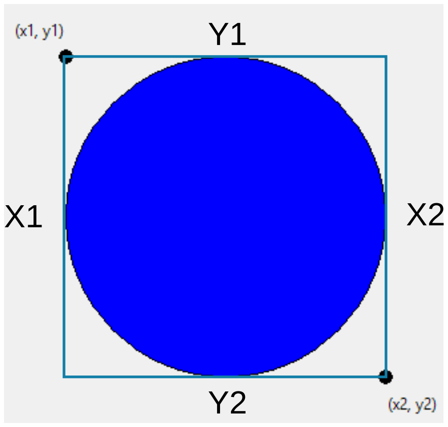
PART 2: Making the Circle stay on Screen

Next, we'll be adding code to make sure our blue circle can't be moved off screen.

We will need to add a conditional statement to our `movement()` function that will not allow the circle to move in a direction if it will move them off the screen.

So if we are on the left edge of the screen and the user presses the left arrow key, we need to make sure we don't move the circle. This is the same for all other directions.

Remember that a circle object has four coordinates. The x1 and y1 coordinate of the top left corner and the x2 and y2 coordinate of the bottom right corner.



We can get these coordinates by using the function `canvas.coords(c)` which returns a list of the four coordinates (x1, y1, x2, y2).

Try adding the code INSIDE our movement function:

```
def movement():
    pos = canvas.coords(c)  #get player coordinates
    if (moveleft and int(pos[0]) >= 3):  #if x1 > 3
        canvas.move(c, -3, 0)
    if (moveright and int(pos[2]) <= 397):  #if x2 < 397
         canvas.move(c, 3, 0)
    if (moveup and int(pos[1]) >= 3):   #if y1 > 3
        canvas.move(c, 0, -3)
    if (movedown and int(pos[3]) <= 397):  #if y2 < 397
        canvas.move(c, 0, speed)
```

So we get the circle's four coordinates and store it in the variable `pos`.

Each of the coordinates in `pos` will let us know if we are close to any of the screen edges, so we have to use them all for the correct direction. We also make sure to convert them to an int because they are given as a float.

Test your code here. You should notice that we cannot move our circle off the screen anymore.

PART 3: Creating a Timer

Our game will need to keep time for how long we've played. This is really easy.

We need two `float` variables: one for the total seconds and one for the maximum seconds. Once the total seconds is greater than the maximum, end the game.

Try adding this line ABOVE your main loop:

```
c = canvas.create_oval(180, 180, 220, 220, fill="blue")
totalsec = 0.0      #keep track of total seconds
maxsec = 60.0       #maximum seconds to play (60 secs or 1 min)
while True:
```

Now, we need to add some time for every loop and check if our total is greater than the max. We will need to add some code in our game loop.

Look at our game loop right now and remember that `time.sleep(0.01)` means that we wait 0.01 seconds every loop. So we can simple add 0.01 seconds to the total seconds every time we get in the loop. Then we need to check if our total seconds is greater than our max amount of seconds.

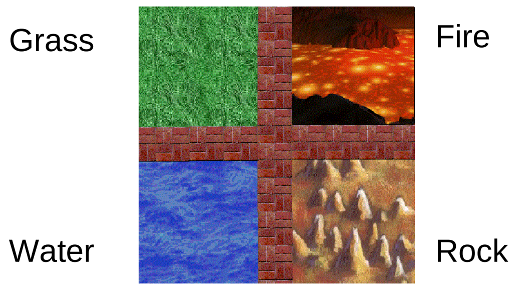
Try adding the following code INSIDE your main loop and fill in the blank:

```
while True:

    movement()

    myTk.update()   # Call the myTk update function

    time.sleep(0.01)   # Wait before drawing the next frame

    totalsec +=__________#add wait time to total secs

    if(_____________):   #check if our total >= max

        break            #break out of while loop and end game
```

Now we can test our code. Make sure you go back into the code and make `maxsec` equal to a smaller number or else you will need to wait a minute to see if the window closes. I would make `maxsec = 5.0`, run the code, and then count to 5 seconds slowly and see if the screen closes in time. After you're done, make `maxsec = 60.0` again.

PART 4: Setting up our Game Background

We will now add a background image called "pokemon_map_v2.gif" that looks like this:



It will have four different terrains, separated by a brick road. Later on we will be adding certain pokemon to each terrain, so that the player needs to search each terrain if they want to catch them all.

Try adding the following code ABOVE creating the circle:

```
canvas.pack()   # Pack the canvas onto the screen
background = PhotoImage(file="pokemon_map_v2.gif")
canvas.create_image(0, 0, anchor=NW, image=background)
c = canvas.create_oval(180, 180, 220, 220, fill="blue")
```

To place an image on a canvas you need to first save a `PhotoImage` variable and then use it with the `canvas.create_image()` function.

```
canvas.create_image(0, 0, anchor=NW, image=background)
```

This code will place the image `background` on the canvas with its northwest (NW) corner anchored at (0, 0). The image size is 400 x 400 pixels so it will cover the screen.

PART 5: Running into Pokemon Randomly

In this part we will be simulating a random pokemon encounter by running a function called encounter() every time we move.

Inside the function encounter, we will randomly see if we have run into a pokemon.

Try adding the following code ABOVE the other functions:

```
c = canvas.create_oval(180, 180, 220, 220, fill="blue")
def encounter():
    encountered = False
    encounter_probability = random.randint(1, 100)
    if(encounter_probability == 1):  #A 1/100 chance of encounter
         encountered = True
    if(encountered):
         print(“You ran into a pokemon!”)
def keypress(event):
```

The code will choose a random number between 1 and 100, and if it happens to be 1 then we have encountered a pokemon. Now we just need to call this function every time we move, by placing it in our `movement()` function.

Try adding the following code INSIDE the movement function:

```
def movement():
    pos = canvas.coords(c)
    if (moveleft and int(pos[0]) >= 3):
        canvas.move(c, -3, 0)
        encounter()       #Do this three more times for all moves
    if (moveright and int(pos[2]) <= 397):
    . . .
```

Now the `encounter()` function will run every time the circle moves, which can happen every 1/100th of a second in the main loop. And the encounter function will randomly choose a number that has a 1/100th of a chance of being correct. That means there should be an average encounter every second as long as the circle is moving.

Before you can test this, make sure you go to the top of the code and import the random module, or else it will crash because we are using random without importing it.

Add this line BELOW the module imports at the top of the code:

```
from tkinter import *
import time
import random       #need this module to use random
```

Run your code and see if the shell prints out our encounter message when you start moving. If you think the messages are getting printed too often, then try changing

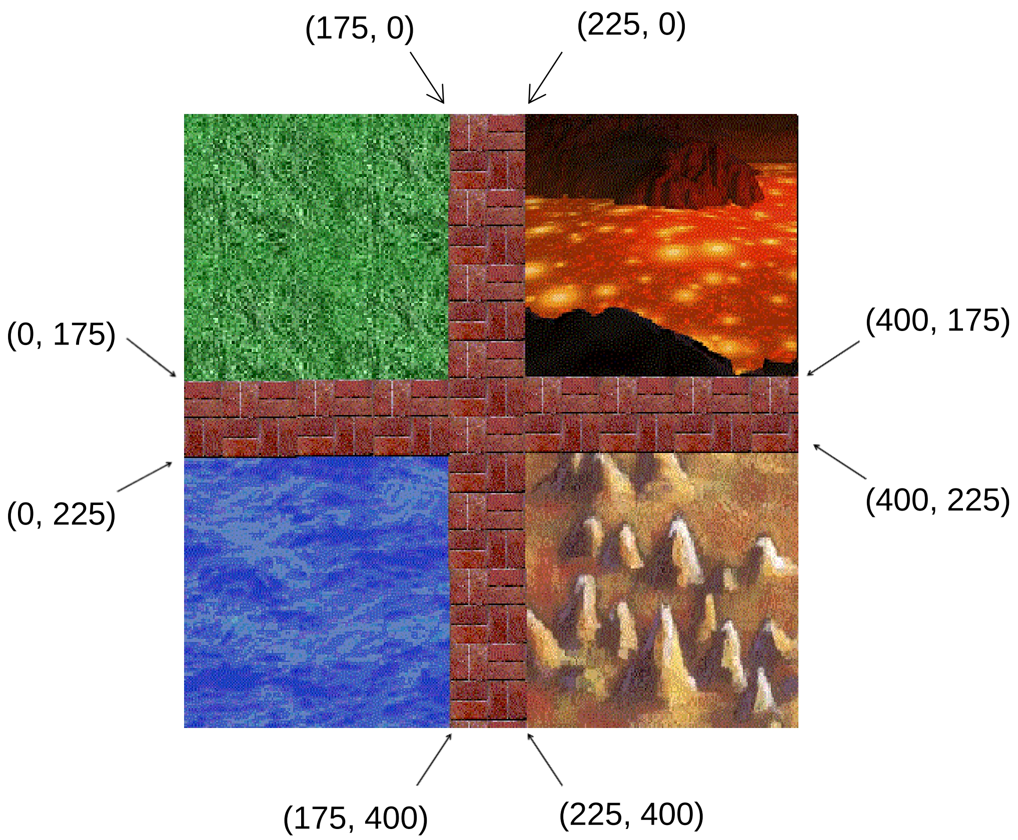
```
encounter_probability = random.randint(1, 200)
```

Run your code again. This should make the encounter happen an average of every two seconds when moving.

PART 6: Limiting Encounters to only Terrain

In this part, we will be adding code so that an encounter will only happen if the circle is on terrain. We don't want an encounter if we are on the brick road. This will be very similar to the logic we used for making sure the circle could not move off the screen.

It's important to know the screen is 400 x 400 pixels, the roads make a + symbol in the center of the screen at (200, 200), and the road is 50 pixels wide.



Add these lines INSIDE the encounter function and fill in the blanks:

```
def encounter():
    . . .
    pos = _____________    #get circle coordinates from canvas
    if(encountered):       #Hint: it’s either 175 or 225
        if(int(pos[2]) <= 175 and int(pos[3]) <= 175):
            print(“You ran into a grass pokemon!”)
        if(int(pos[2]) <= 175 and int(pos[1]) >= ___):
            print(“You ran into a water pokemon!”)
        if(int(pos[0]) >= 225 and int(pos[1]) >= 225):
            print(“You ran into a rock pokemon!”)
        if(int(pos[0]) >= ___ and int(pos[3]) <= ___):
            print(“You ran into a fire pokemon!”)
```

Once you finish, try running your code. Test it out by walking on the road only. There shouldn’t be any print messages. If there is, then recheck the numbers you filled in.

If you don’t get any messages on the brick road while moving, then try moving into each of the terrains and make sure you get a pokemon from each terrain.

If all the terrain messages work, then good job! The game is more than halfway done.

PART 7: Spawning Random Pokemon by Terrain

In this part, we will be spawning random pokemon when we get an encounter.

We need to add four arrays with three pokemon for each terrain. We also need to add print statements in the beginning to give the players instructions and game information.

Add these lines ABOVE the encounter function:

```
moveup = False


found = []          #this array will hold the pokemon we find

fire = ["Charmander", "Magmar", “Vulpix"]

water = ["Squirtle", "Goldeen", "Magicarp"]

rock = ["Onix", "Sandshrew", "Geodude"]

grass = ["Weedle", "Caterpie", "Oddish"]


print("Welcome to Pokemon Safari")

print("Use arrow keys to explore the 4 different terrains")

print("There are 3 different Pokemon for every terrain")

print("Try to find and catch all 12 before time runs out!")


def encounter():
```

Run the code. Our game should start with a welcome message.

Notice that we make an empty array called `found`, which we’ll use later to track how many pokemon we found.

Now we need to make a function that will accept a terrain array (ex. fire), and will spawn a random pokemon from that array (ex. charmander) when we have an encounter.

This function will replace all the print statements in our `encounter()` function and will print out that we found a specific pokemon from that terrain.

Add these lines ABOVE the encounter function:

```
print("Try to find and catch all 12 before time runs out!")


def spawn(terrain):  #make new function accepting an array
    pokemon = random.choice(terrain) #pick random from array
    print("You found", pokemon)   #print out we found the pokemon
    if(pokemon not in found): #if we haven't found it before
         found.append(pokemon) #add pokemon to list
def encounter():
```

Note that we are adding `pokemon` to our `found` list only if we haven't found it before.

Now that we have a function that spawns a random pokemon based on the terrain, we only need to call it from the `encounter()` function.

Replace the print messages in `encounter()` with the `spawn(terrain)` function that has the appropriate terrain.

Change these lines INSIDE the encounter function:

```
if(encountered):
    if(int(pos[2]) <= 175 and int(pos[3]) <= 175):
        ~~print("You ran into a grass pokemon!")~~ #Delete this
        spawn(grass)  #Do this three more times for all terrain
    if(int(pos[2]) <= 175 and int(pos[1]) >= 225):
       . . .
```

Once you finish changing these lines, run your code and test it by running to each terrain.

You should be seeing print messages saying you found a certain pokemon from that terrain.

In order to win the game, you need to ***catch*** all the pokemon and not just ***find*** them. If you haven't played Pokemon before, the way to catch them is throw a pokeball at them when you find them, and hope it they stay inside the pokeball. For our game, we will be flipping a coin (tails or heads) to see if the player catches the pokemon they find or not.

PART 8: Catching the Pokemon

In this part we will add code so that we can try to catch the pokemon we find.

We need to add another empty array like `found`, but call it `caught`. This array will keep track of how many unique pokemon we catch.

Add this line ABOVE the terrain arrays:

```
found = []          #this array will hold the pokemon we find
caught = []         #this array will hold the pokemon we catch
fire = ["charmander", "magmar", "vulpix"]
. . .
```

Now we need to flip a coin when we find a pokemon, and either we catch the pokemon, or the pokemon runs away. This will happen in our `spawn` function.

Add these lines INSIDE the spawn function and fill in the blank:

```
def spawn(terrain):
    . . .
    if(pokemon not in found):   #if we haven’t found before
         found.append(pokemon) #add pokemon to found list
    coinflip = random.randint(0, 1) #heads or tails
    if(coinflip == 0): #heads means pokemon ran away
         print(pokemon, “ran away”)
    else:              #tails means we caught the pokemon
         print(“You caught”, pokemon)
         if(pokemon not in caught):#if we haven’t caught before
              ____________________#add pokemon to caught list
def encounter():
```

Once you finish, test your code. You should now see a second message when we find a pokemon. Either we catch it or the pokemon ran away.

PART 9: Winning and Losing the Game

The final part of this game is to end game once we find and catch all the pokemon.

This will happen in our main loop, where we end the game if our time’s up.

Add these lines INSIDE the main loop:

```
 while True:

     . . .

     if(totalsec >= maxsec): #If time is up

          print(“Time’s Up! Game Over!”) #Lose message

          break

     elif(len(caught) == 12): #If we caught 12 pokemon

          print(“Congrats! You caught them all!”) #Win Message

          break

#Print out the final count for the found/caught pokemon

print("You found " + str(len(found)) + " of the 12 pokemon")

if(len(found) > 0):   #if we found at least 1 pokemon

    print ("Found pokemon are: " + str(found)) #print found list

print("You caught " + str(len(caught)) + " of the 12 pokemon")

if(len(caught) > 0):  #if we caught at least 1 pokemon

    print ("Caught pokemon are: " + str(caught)) #print caught list


 myTk.destroy()
```

Something you may not have seen before is `len(array)`, which returns the length of the array. So if `len(caught) == 12` that means we have 12 pokemon in our caught list.

After we print a win or lose message, we break out of the game loop. Then we print a final count of found and caught pokemon and which pokemon where found or caught.

Test out the game and see if you can win. Although it’s possible to win, you may find out that it’s really hard to know which pokemon you caught and which pokemon you still need to catch with all the print messages being outputted. We will fix this in the next part.

PART 10: Bonus: Making the Game More Visually Pleasing

The game may be complete, but it's really hard to win because it's almost impossible to keep track of which pokemon you still need to catch.

To fix this, we will be getting rid of most our print statements and having them show up as text on the screen. We will only print when we catch a new pokemon.

To do this we will be doing a lot of changes in the `spawn` function and be using canvas methods like `create_text()`, `create_rectangle()`, `tag_lower()`, and `delete()`.

What we want to happen is when we find a pokemon, a text box pops up on the screen saying we found a pokemon. Then wait half a second. Flip the coin and see if we caught the pokemon. The we remove the previous text box and show a new text box saying if we caught the pokemon or it ran away. Wait another half second. Then delete the text box and resume play.

In order to wait a half second, we will be using the function `time.sleep()` again. The reason we want to wait a half second is so that the player can read the text box before we delete it and is not able to catch more pokemon while the old text box is still showing.

Also, it's important to note that if we wait a second outside our main loop, then it will not add a second to our `totalsec` variable. Basically, our game clock will stop when we wait.

We will first create a new function called `temporary_text(message)`, which will show a text box with the message, wait a half second, and then delete the text.

Add the following code ABOVE the spawn function:

```
def temporary_text(message):
    textbox = canvas.create_text(200, 200, font=("Purisa", 16), text=message)
    rect = canvas.create_rectangle(canvas.bbox(textbox),fill="white")
    canvas.tag_lower(rect,textbox)
    myTk.update()   # Call the myTk update function
    time.sleep(0.5)   #wait half a second
    canvas.delete(textbox)    #delete text
    canvas.delete(rect)    #delete white rectangle
def spawn(terrain):
```

`canvas.bbox(textbox)` returns the bounding box of the textbox which is four coordinates of the top left and bottom right corners, so we can place a white rectangle background the same size and place as the text. This is like `canvas.coords(c)` that returned the four coordinates of the circle. The reason we can't use `coords()` right now is because a text object will only return two coordinates, and the rectangle needs four coordinates in order to be properly created.

We also use `canvas.tag_lower(rect, textbox)` which makes sure our white rectangle background will be behind our textbox.

Now we just need to use this function, instead of printing when we find and catch pokemon. We will also need to add a print message when we catch a new pokemon, so the player clearly knows how many they caught and which pokemon they caught as they're playing. This will all be changes in the spawn function.

Change the following code INSIDE the spawn function:

```
def spawn(terrain)
     pokemon = random.choice(terrain)
     text = "You found " + pokemon       #replace print() here
     temporary_text(text)
     if(pokemon not in found):
          found.append(pokemon)
     coinflip = random.randint(0, 1)
     if(coinflip == 0):
          text = pokemon + " ran away"   #replace print() here
          temporary_text(text)
     else:
          text = "You caught " + pokemon #replace print() here
          temporary_text(text)
     if(pokemon not in caught):
          caught.append(pokemon)
          print(len(caught), text) #print newly caught pokemon
def encounter:
```

Now test your game. If everything is working, you should have messages popping up in the middle of the screen letting you know if you find a pokemon, and whether you successfully caught it or not. Also, there should not be a large amount of printing in your shell. It should print the beginning message, print only when you catch new pokemon, and then the final message once you win or lose the game.

There is still a lot that you can do to make this game better, but for now this lesson is over. Some ideas you may want to try on your own are:

- Show a timer and score of how many pokemon caught on the screen
- Rename the pokemon to other pokemon names or made up names
- Change one of the terrains by modifying the background in Paint
    - Then make a new set of pokemon for that terrain, like ice for example
- Show a image of the pokemon you find to appear on the screen for half second
- Bind a button, like Enter, that will make the game end early
- Change the difficulty by changing the encounters per second and max seconds