Quevedo Ramos, Brayan

18, April 2022

Professor Beth

**Exam 2: Code Dissection**

***from graphics import \****

The first line of a code is a call to the directory of Python. In this directory, usually the node folder, Python shall scan for the file named “graphics”. If Python is able to pop “graphics”, it does one of two things (from trial and error): one, it stashes all the code in static memory. Every line has floating memory value that’s accessible given it’s a public entity and stored upon execution. Two, Python only pops defined function names within the program. While more reasonable as it’s less resource intensive, if there’s input that aren’t in functions that are required as they’re parameters in other functions (such as in a dynamic function), it would crash the program.

***import time***

This line of code calls for the “time” library in Python. One can safely assume that time is a keyword in Python, so instead of doing a full scope scan, this improving performance.

***#create a window, specify title, width, height***

A comment telling the user what the below line(s) of code shall perform. This code will not be executed at runtime.

***win = GraphWin("My First Graphics Program!!!", 400, 400)***

This line of code creates a variable in memory called “win”. “win” is defined by calling the graphics library (which is stored in memory) and calls the function \_\_init\_\_ in the “GraphWin” class. “GraphWin” creates a graphics window in Python, but needs the following parameters: str, int, int. Finally, “win” shall be stored in memory.

***#create a point by specifying the x and y positions***

A comment telling the user what the below line(s) of code shall perform. This code will not be executed at runtime.

***point1 = Point(200, 200)***

This line of code creates a variable in local memory called “point1.” “point1” is an instance of an object (Point) called from the Graphics library. In order to pass the \_\_init\_\_ stage of objects, it needs two parameters: int, int, to which this line of code complies. Point stores an x and y coordinate in memory of where something will happen on the graphics window.

***#create circle, specify the center point and radius***

A comment telling the user what the below line(s) of code shall perform. This code will not be executed at runtime.

***image1 = Image(point1, "C://Users//Betty//Desktop//Rabbit.gif")***

This line of code creates an instance of the object “Image” where it contains two parameters in the \_\_init\_\_ functions, a point (which is a defined instance of an object such as point1), and a str (which is a file path to search for a specified file). Then it stores all of this in memory. Once called, image1 will display an image inside of the graphics window at a given point. More specifically, the center of the image will be the location of the image will be displayed at the specific coordinate point.

***#make it show up in the window***

A comment telling the user what the below line(s) of code shall perform. This code will not be executed at runtime.

***image1.draw(win)***

This line of code is really cool as it demonstrates what’s fantastic about OOP – functions in objects. In this instance, we grab image1 (and it’s properties), and win (with its properties). The instance of Image1 has a class function named “draw” to make the object be displayed in some window. The window is already defined as an instance in local memory via “win.”

***speedX = 10***

Define an integer variable named speedX and give It a value of 10. Save in memory.

***speedY = 7***

Define an integer variable named SpeedY and give it a value of 7. Save in memory.

***#slide the circle over in the x direction one pixel at a time***

A comment telling the user what the below line(s) of code shall perform. This code will not be executed at runtime.

***while True:***

Create a loop where the condition must be changed to break. In this case, since True allows things to happen, the indented lines will continue to be ran until broken.

***centerPoint = image1.getAnchor()***

Create a local variable named centerPoint. This variable will be result of the getAnchor class function where the function gets cloned via: the instance (from image1) of an instance (image). This will be used to get the coordinates of the variables listed below.

***xPos = centerPoint.getX()***

Create a local variable named xPos from the class centerPoint. This class has a class function named getX() that shall return the x-coordinate of an object to store.

***yPos = centerPoint.getY()***

Create a local variable named yPos from the class centerPoint. This class has a class function named getY() that shall return the y-coordinate of an object to store.

***if (xPos + (image1.getWidth() / 2)) > 400 or (xPos- (image1.getWidth() / 2)) < 0:***

This if statement is quite tricky as there’s lots of moving parts. It has two parts, the left side and right side of the or statement. The left side sees if the image is in bounds per defined graphics window. This is so the entire image is able to be captured on the window without fallout. The second part of the or makes sure that the width of the image is NOT less than 0, as it’s impossible to have negative width of an image (how would that work?)

***speedX = speedX \* -1***

Redefine the existing variable of speedX as the inverse sign of it’s (now) previous iteration via memory.

***if (yPos + (image1.getHeight() / 2)) > 400 or (yPos - (image1.getHeight() / 2)) < 0:***

This if statement has two parts, the left side and right side of the or statement. The left side sees if the image is in bounds per defined graphics window. This is so the entire image is able to be captured on the window without fallout. The second part of the or makes sure that the height of the image is NOT less than 0, as it’s impossible to have negative height of an image (how would that work?)

***speedY = speedY \* -1***

Redefine the existing variable of speedY as the inverse sign of it’s (now) previous iteration via memory.

***image1.move(speedX, speedY)***

Call the move function from the class image1 (and it’s inherited values). The move function will… move… the specified object (in this instance an image) to a specified X/Y point on the graphics window. The X & Y values needed to move are provided by the int variables: speedX, and speedY.

***time.sleep(0.05)***

Call the sleep function from the library time, with the given parameter of 0.05 float. This means that the program will idle for the given interval in seconds.