| Unit 8 Class & Object Problems [Learning Plan Index - Python](https://docs.google.com/document/d/1Ys_VACt7p2P5NUuhDetAiOVoUwDGXNA_h2P2aBKoZ5Y/edit?usp=sharing)    *Unit 07 of Python Programming - Unit 8 Class & Object Problems* | |
| --- | --- |
| Learning Targets  This unit we will…  Explore classes, what are they, how do we create and use them, and how do we create objects with them. We will also learn what a UML diagram is, how to read them and how to create one for our classes.  I can…   * Create a class with public or private variables. * Create class methods with or without parameters. * Use constructor methods to set values for class variables when creating instances of the class, known as instantiation. * Use getter and setter methods to get and set class variables. * Use objects as variables of a class, which is composition. * Create programs that create and use class objects.   Vocabulary: Class, object, instantiation, constructor, class variables, instance variables, encapsulation, private variables, public variables, methods, getters, setters, UML ([Unified Modeling Language](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/)) | |
| Learn About It!  *You can explore some, or all of these resources. If you want to see a resource again, go for it!*  [Learning Plan Index - Python](https://docs.google.com/document/d/1Ys_VACt7p2P5NUuhDetAiOVoUwDGXNA_h2P2aBKoZ5Y/edit?usp=sharing) *These Collab documents review the concepts of each unit with code you can run and modify.* | |
| Evidence of Learning  *Complete the following programming exercises.*  [Grading Rubric](https://docs.google.com/document/d/1E-sYsO3jqyfYSy0Yu4c_pWyOi4HEFA5Z3NFEAeyoT4M/edit?usp=sharing) | |
| Unit Programs  Review:   1. [Colab - Classes and Objects](https://colab.research.google.com/drive/1_HVM7vHMi50l4fGnex9v4PjRxbGHQ6Dx) 2. [08P - 02 - UML Student Diagram Copy](https://docs.google.com/document/d/1ufuKpSJrJ9POLUp94BFYrm_lF7KegySpo3gEXvhKqEM/edit?usp=sharing) 3. [Sample Circle Class Repl.it](https://replit.com/@MrReynolds/Chap7-Circle-Sample#main.py) 4. [Sample Time from time.time()](https://replit.com/@MrReynolds/Unit08ShowCurrentTime#main.py)   Once you have reviewed the Colab document, complete the problems below. There are tips, sample code, and links to sample code that you will use within the Colab documents, you also may want to refer back to early colabs. This project has special directions below about the naming of files and will give you links to some starting code with the proper file naming structure. If you work in pycharm you will need to zip your final files and turn in the one zipped file when you are done.  **Lab Overview/Directions/Starting Code**  This set of problems is going to require a number of subfiles either in repl.it or in pycharm. Your class code will go in the subfiles and will be imported into your main file for use when you use the classes to create objects to solve the problems. If you look at [Sample Circle Class Repl.it](https://replit.com/@MrReynolds/Chap7-Circle-Sample#main.py) you will see that there is more then one file on the left hand side and at the top of the main.py file the classes from those files are imported into main with the following lines of code:  from Circle import Circle  from CirclePrivate import CirclePrivate  There is a [starting version of this project in repl.it](https://replit.com/@MrReynolds/Unit08StudentStart#main.py) you can fork this repl.it for this project or create a new project in pycharm and then create separate files with the same names that the repl.it has and copy all the code from the repl.it files over to your files. You will also notice in main.py in the starting file that the first problem has already had the Rectangle and RectanglePrivate classes imported into it. Also the problem code has been placed in a function and then the function gets called right after. This way you can comment out the function call to turn one of the problems off. This same approach is used in the sample circle repl.it. You are required to put each one of your solutions in a function so it can be easily deactivated. Keep in mind in repl.it when you hit run the only file that gets run is the main file. So the code in your sub class files will not get run until they are called with an import statement in the code inside the main file. Below I am going to break down each problem separately, but all the solutions should be in your main.py file while all of your class code should be in your class files. This is a more challenging and complex lab so please ask questions if you have them.  **Rectangle & RectanglePrivate Classes**  **Sample output is shown below**   1. Rectangle & RectanglePrivate Classes - (15 points) -  You will need to complete the UML ([Unified Modeling Language](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/)) diagram for RectanglePrivate in the Google Doc UML diagram copy that was made for you in the classroom (5 of the 15 points). The other UML Diagrams have been provided to you to help you understand what variables and methods the classes should have. You can look at the two versions of the Circle classes for some guidance. The big difference between the two rectangle classes is that RectanglePrivate needs to have private variables. This change will require two more methods: a setter and getter method. You may want to complete the code for this problem before completing the UML Diagram. 2. In the Rectangle.py file you need to declare a Rectangle class. The constructor (aka: the initialize or init method, or \_\_init\_\_) needs to take a width with a default value of 1 and a height with a default value of 2. If you look at the sample Circle file you will see that it has a private variable radius with a default value of 1: class CirclePrivate:  def \_\_init\_\_(self, radius = 1.0):  self.radius = radius 3. Next you will need to create a getWidth() method that returns the width value, look at the sample circle class and the getRadius() method for an example. You also need to create a getHeight() method that returns the height. 4. Now you need to create a getPerimeter() method that returns the value of the perimeter of the rectangle using the class height and width variables. You also need a getArea() method that returns the area of the rectangle again using the class height and width variables. Look at the circle class and the getPerimeter() and getArea() methods of those classes as a model for what you need to do. 5. The last method you need to create is the \_\_str\_\_ method. This method defines what should be output, as a String, of an object when the object is printed. Without this method if an object is printed the memory location and type of the object will be output. Again look to the \_\_str\_\_ method in the circle class as a model. Also look at the sample output for this problem below to see how the output of this method should look. You need to output the width, height, area, and perimeter of the circle. You have the code in main to create a rectangle with this class and print it so if you run your program you should get the following output:  Notice there are two lines of output because there are two different print statements in main. One manually concatenates all the different outputs into one print statement and the other just says print object. I have done this so you can see the advantage of the \_\_str\_\_ method for when you need to print objects, it makes it very convenient. Not all objects lend themselves to being printed, but when they do, having this extra method can be very handy, as you’ll see in these exercises. 6. Now copy your Rectangle code into the RectanglePrivate file and change the variables to private, look at CirclePrivate for guidance. Make sure that you have changed the variables to private wherever they are accessed within the class. 7. Now that you have private variables you need to create a setWidth() method that will have a parameter that will be assigned to the class width variable, look at the sample CirclePrivate setRadius() method for an example. Next you need to create a setHeight() method with a parameter for changing the class height variable. In your main is three lines of code that create and print a RectanglePrivate, uncomment those lines to see if you get this output: 8. Now that you have two working classes here is what I want you to do in main, in the prob1() function after the existing code:    1. print(“Change width of public rectangle variable directly to 20”) and then access the public variable directly and change it to 20, rect1.width = 20. Then print rect1.    2. print(“Try to change the width of the private rectangle variable directly to 70”) and try to change the width of rect2 directly to 70 like you did with rect1 and the print rect2.    3. print("Change width variable with set method to 25.0") and use the rect2.setWidth() method to change the width to 25.0 and print rect2. 9. Your finished output for this problem should look like this:   **Account Class Sample output is shown below**   1. Account Class - (20 points) -  In the Account class I have given you some code to start with, the class declaration, the constructor method and the \_\_str\_\_, or toString method. I gave you this code because in this class the Time class is used to set the account id to a unique integer. This isn’t a totally practical application, but demonstrates the idea that when dealing with records we would want unique ID numbers to identify the different accounts. In reality a number like this would be created and managed by database software. 2. You need to add a number of methods to this class, refer to the [UML diagram](https://docs.google.com/document/d/1KfyXrXqX4yUPPDQv2jxjOi44ivnrTnYzonhpT99w4SU/edit?usp=sharing) for reference.    1. getID() will return the idd number of the object.    2. getName() will return the value of the name variable.    3. getChecking() will return the value of the checking variable.    4. getSavings() will return the value of the savings variable.    5. checkingDeposit(deposit) needs to first check that the deposit parameter is positive, return a message if it isn’t, and if it is it needs to add that value to the checking variable. After a deposit there should be a return statement stating that the deposit amount was successfully deposited, likewise if the deposit parameter is negative a return statement should output that the deposit needs to be a positive value. Look at the sample output for example output.    6. checkingWithdrawl(withdrawal) will first need to check that the withdrawal parameter is positive, return a message if it is not. Next it needs to check if the value in the checking variable is large enough to subtract the withdrawal. If it is, subtract the withdrawal amount from checking and return a message to the user. If checking is not a large enough number, check to see if checking and saving combined is large enough. If it is, make the checking variable 0 and subtract the extra amount from savings, then return a message to the user about the transaction. Finally, if there is not enough money to cover the withdrawal in the two accounts, return a message to the user stating that there are insufficient funds.    7. savingsDeposit(deposit) will work like the checking version, make sure the number is positive and if it is, add that money to the savings variable and return a successful deposit message, if not a positive number return a message about positive numbers.    8. savingsWithdrawal(withdrawal) will work like the checking version, but a little easier. Make sure the withdrawal parameter is positive and return a message if it isn’t. Check to see if the saving variable is large enough for the withdrawal, if it is subtract the amount and return a message. If savings is not large enough, return a message about insufficient funds. 3. Now in your main file in your prob2() function create an account object, pass it your first name, $2000 for checking and $1000 for savings and print your account object. Now do the following.    1. Try and deposit -200 to the checkingDeposit method and make sure your return output is printed to the screen and print your object.    2. Deposit 200 to the checkingDeposit method and make sure your return output is printed to the screen and print your object.    3. Try and withdraw 5000 with your checkingWithdrawal method and make sure your return output is printed to the screen and print your object.    4. Try to withdraw -1000 with your checkingWithdrawal method and make sure your return output is printed to the screen and print your object.    5. Withdraw 500 with your checkingWithdrawal method and make sure your return output is printed to the screen and print your object.    6. Try and deposit -200 to the savingsDeposit method and make sure your return output is printed to the screen and print your object.    7. Deposit 200 to the savingsDeposit method and make sure your return output is printed to the screen and print your object.    8. Try and withdraw 5000 with your savingsWithdrawal method and make sure your return output is printed to the screen and print your object.    9. Withdraw 500 with your savingsWithdrawal method and make sure your return output is printed to the screen and print your object.    10. Withdraw 1800 with your checkingWithdrawal method and make sure your return output is printed to the screen and print your object. 4. Your completed output, when you run your program should look like the output below.   **Student Class Sample output is shown below**   1. Student Class - (20 points) -  I have given you some starting code for the student class because this class imports time and sets the gradYear to the current year as a default. I was going to have code that calculated what class you were in (junior, senior, etc.) based on the current year and your gradYear, but I decided to skip that, you're welcome. I have also given you the printClasses() method, which creates a string output of the student’s classes, their class scores, and their score average for all of their classes. Finally I have given you the \_\_str\_\_ method for printing out your object. So I have basically done all the hard work for this problem, again your welcome … and yes I am lying. 2. Now you need to create the following methods, refer to the [UML diagram](https://docs.google.com/document/d/1KfyXrXqX4yUPPDQv2jxjOi44ivnrTnYzonhpT99w4SU/edit?usp=sharing) for reference:    1. getName() will return the value of the name variable.    2. setName(aname) will allow for the change of the name value to whatever is passed in with the parameter.    3. getGradYear() will return the value of the gradYear variable.    4. setGradYear() will allow for the change of the gradYear value to whatever is passed in with the parameter.    5. getTownr() will return the value of the town variable.    6. setTown() will allow for the change of the town value to whatever is passed in with the parameter.    7. addClass(aclass, score) will allow for a class name and a score value parameter to be passed in, those two items need to be added to the classes and scores list respectively. It is important that these items are added at the same index point in their respective lists, because the indexes need to match for the class and score to be properly associated. At this time we do not have code to account for this, we may need to change this to a two dimensional list in a future update. But not your problem for now. 3. Now in main for prob3() create a student object.    1. Get user input for what the student name should be, enter your name and use setName() to set the name variable to the user input.    2. Get user input for what the student grad year should be, enter your grad year and use setGradYear() to set the gradYear variable to the user input.    3. Get user input for what the student town should be, enter your town and use setTown() to set the town variable to the user input.    4. Create a loop for entering student classes and scores (grades in a numerical format i.e. 70). In the loop get the user to enter a class name and then, with a second input get the score for that class. Once you have the two values pass them into the addClass(aclass, score) method which will add the values to the class and score lists. After the method call, ask the user if they want to enter another class and score. Keep adding classes and scores as long as the user indicates that’s what they want. Exit the loop when the user is done. Enter your current classes and a made up score for each class.    5. Once the loop is complete print the student object. Your output should look similar to the output below:   **StopWatch Class Sample output is shown below**   1. StopWatch Class - (20 points) -  For StopWatch I have given you nothing, because I am a mean person. For this class look at the [UML diagram](https://docs.google.com/document/d/1KfyXrXqX4yUPPDQv2jxjOi44ivnrTnYzonhpT99w4SU/edit?usp=sharing) for reference: 2. You will need to import time into your class file. 3. Your \_\_init\_\_ method will need to declare two private variables startTime and endTime, they will be given a default value of 0 in the parameter list, otherwise they will be given the value passed into the parameter. 4. The start() method will set the startTime variable to time.time(). 5. The stop() method will set the endTime variable to time.time(). 6. For this next method refer to the [time repl.it](https://replit.com/@MrReynolds/Unit08ShowCurrentTime#main.py) that was shared earlier in the learning plan. The returnTime(thetime) method will take a time parameter, convert it to an int and then pull the following variables from the time input: currentHours, currentMinutes, currentSeconds, and currentMilliseconds. The method will then return a String representation of the time in the format: currentHours:currentMinutes:currentSeconds:currentMilliseconds. Keep in mind you will have to adjust the hour to the eastern time zone, getting military time is fine. 7. The method getStartTime() will return a call to returnTime(startTime). 8. The method getEndTime() will return a call to returnTime(endTime). 9. The method getElapsed() will return a call to returnTime( endTime - startTime). 10. In main, prob4() create a clock object and call the start() method. Have a for loop that goes from 1-1,000,000 while summing all the loop counters of the loop. When the loop is done, call the stop() method. 11. Output the following similar to the output below:     1. The start time from getStartTime().     2. Sum of the loop formatted with commas where needed in the number     3. The elapsed time from getElapsed().     4. The stop time from getEndTime() 12. Start the clock again and have a for loop that counts down from 10 to 1 with a 1 second pause between each number while outputting the countdown. At the end of the loop stop() the clock and output:     1. The start time from getStartTime().     2. The elapsed time from getElapsed().     3. The stop time from getEndTime() 13. Your final output should look like this:   **Line & LinearEquation Classes Sample output is shown below**   1. Line & LinearEquation Classes - (25 points) - This problem involves two classes. The line class will be used for creating two line objects, which will be passed into the LinearEquation class in order to create a LinearEquation object that can be used to calculate the intersection point of the two lines so that a small circle can be drawn on that intersection point. A lot of the required code has been given to you, but there is still a lot of code that needs to be completed and there is a bug that has to be dealt with. 2. In the Line class complete the drawLine(acolor) method. This method will take a color as a parameter that will be used to set the color of the line to be drawn. Using the turtle class variable, set the color, pick the pen up, go to one of the line points, put the pen down and go to the other line point, then pick the pen up again. Do the following line methods:     1. For the getDeltaY() method use the slope formula to return the difference (delta) value of the two Y values of the line.    2. For the getDeltaX() method use the slope formula to return the difference (delta) value of the two X values of the line.    3. For the getM() method use the getDeltaY() and getDeltaX() methods to return the slope of the line.    4. For the getB() method use the starting X & Y values and getM() to return the y-intercept value of the line. 3. For the Linear equation you will pass two lines in when you create the object. You need to complete the following methods.    1. drawGraph() this really doesn’t belong in this class, but when called this method will, from (0,0) using the class turtle variable, draw the cartesian coordinate system from -300 - 300 for both the X and Y axis. This method will get called once after the first LinearEquation object is created and then all the other equations will draw their lines on the one graph.    2. drawVertex() will use the getX() and getY() methods to get the X & Y coordinates of the intersecting point of the two lines and it will draw a 2 pixel solid red circle with a black outline around the intersecting point. There will be a bug in this method that you will have to manage.    3. getX() using the equation below return the x value of the intersecting point of the two lines. This method will have the same bug as the drawVertex() that you will have to manage.    4. getY() will use the slope intercept formula to get the y value of the intersection point of the two lines. 4. Once your two classes are complete in main in your prob5() function do the following:    1. Create two line objects:       1. line1 = Line(200,200,0,0)       2. line2 = Line(0,200,200,0)    2. Create a LinearEquation object passing in line1 & line2    3. Use the LinearEquation object to draw the cartesian coordinate system with the drawGraph() method. ONLY DRAW THE GRAPH ONCE.    4. Draw line1 & line2, setting them to [two different colors](https://trinket.io/docs/colors).    5. Print line1 & line2 and print line1 & line2 value for m as well as their value for b. These are six different print statements; see the sample output below.    6. Draw the circle around the intersecting point of the two lines with the drawVertex() method.    7. You will repeat the above process a-f (excluding c) for the next three sets of lines    8. Second set of lines:       1. line3 = Line(-250,250,0,0)       2. line4 = Line(-250,0,0,250)    9. Third set of lines:       1. line5 = Line(100,0,0,-100)       2. line6 = Line(0,-10,200,-200)    10. Fourth and final set of lines:        1. line7 = Line(-100,0,0,-100)        2. line8 = Line(-200,0,0,-200)    11. Sample output is below:       Make sure you have a comment block at the top of your program with your name, the date and a list of the programs that are being run in the program. Also make sure to comment your variables, control structures, and each problem. Also use white space between the problems.  ############################################################  # Name : Date: #  # Unit 8 Problems #  # Rectangle & RectanglePrivate, Account, Student #  # StopWatch, Line & LinearEquation #  ############################################################  When your code works and is commented, turn it into the classroom. | |



Although they mean Class and we are talking about Class.