|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 09/06/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 13B. Using Classes and Composition Objects in Python**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| Muhammad Abubakar Saif | BSCE21017 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to observe the basic knowledge of programming classes in python.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage collected.

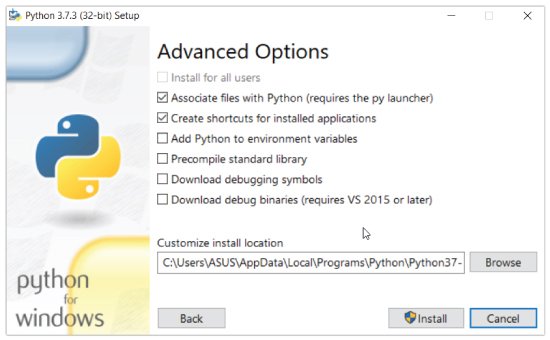
In composition one of the classes is composed of one or more instance of other classes. In other words one class is container and other class is content and if you delete the container object then all of its contents objects are also deleted.

**Lab Task**

**Task A: Run your Program in Python**

1. Download the [latest version of Python](https://www.python.org/downloads/)..
2. Run the installer file and follow the steps to install Python
3. During the install process, check **Add Python to environment variables**. This will add Python to environment variables, and you can run Python from any part of the computer.

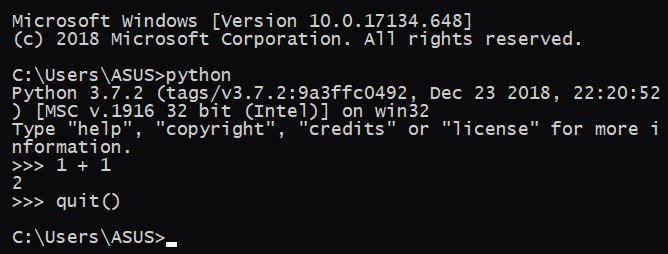
Also, you can choose the path where Python is installed.



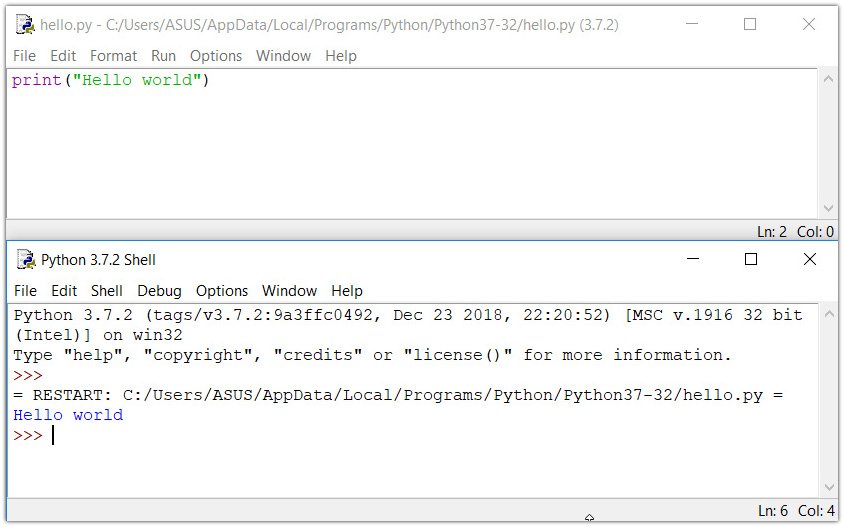
Once you finish the installation process, you can run Python.

1. Once Python is installed, typing python in the command line will invoke the interpreter in immediate mode. We can directly type in Python code, and press Enter to get the output.

Try typing in 1 + 1 and press enter. We get 2 as the output. This prompt can be used as a calculator. To exit this mode, type quit() and press enter.



1. Now you can create a new file and save it with .py extension. For example, **hello.py**



1. Add path of file in windows cmd using cd command, type hello.py and press enter. It will run your python script.

**Task B: Composition [Marks: 40]**

In this task, you are required to create two classes’ **Employee** and **Salary** with the following data members and member functions,

***Private Data Members of class Salary such as:***

pay (int)

***Public Member Functions of class Salary such as:***

**get\_annual\_salary –** It will calculate pay of 12 months and return total annual salary without bonus.

***Public Data Member of class Employee such as:***

pay, bonus (int)

***Public Member Functions of class Employee such as:***

**Constructor() –** It take initialize pay.

**salary\_with\_bonus() –** It will calculate and return total annual salary plus bonus.

Do the following operations in main function:

1. Create object of class **Employee** and call member function **salary\_with\_bonus()**.

|  |
| --- |
| class Salary:  ### Constructor ###  def \_\_init\_\_(self,z):  self.\_\_pay = z #initializing private variable    ### Function for Calculating Annual Salary ###  def get\_annual\_salary(self):  y = 12 \* self.\_\_pay #calculating annual salary  return y      class Employee:  ### Constructor ###  def \_\_init\_\_(self):  self.pay = float(input("Enter your Monthly Salary: "))  self.obj = Salary(self.pay) #Composing Salary Class in Employee Class  self.bonus = 0 #sets member variable to ZERO    ### Function for Calculating Annual Salary with Bonus ###  def salary\_with\_bonus(self):  self.bonus = float (input("Enter your annual bonus: ")) #taking input  return self.obj.get\_annual\_salary() + self.bonus #calculating      ### Main Function ###  obj = Employee() #declaring object of class Emplyee  print(f"Your Annual Salary With Bonus: {obj.salary\_with\_bonus()} $") # calling function and displaying its value s  del obj  ### END ### |

**Output:**

**Text

Description automatically generated**

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & GitHub Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_