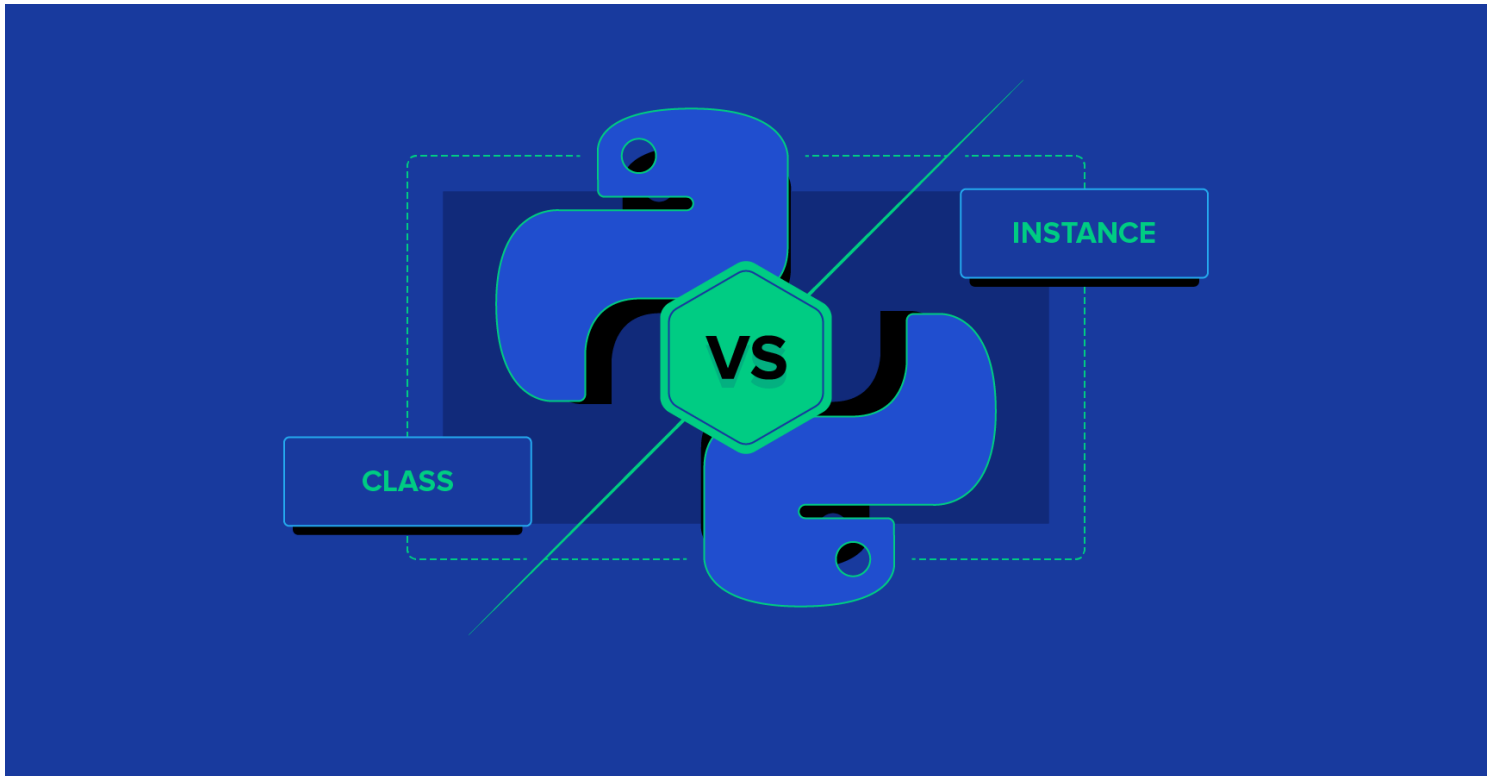


Lab Assignment 4 - OOP

[Start Assignment](#)

Due Tuesday by 11:59pm **Points** 20 **Submitting** a file upload **File Types** py
Available Oct 11 at 8am - Oct 19 at 11:59pm

Objects and Classes



Creating an e-mail Message Class

e-mail. How we often communicate today...from Christmas lists, to pick up from the store lists, to-do lists, to one-liner messages...all examples of communication using an e-mail message.

Lab Assignment Objectives

1. Understand what is meant by a class object's type and how the type determines the operators and methods that can be applied to the object.
2. Know what a constructor does.
3. Understand how a class corresponds to a namespace such that object method invocations translate to method calls in the same namespace.

4. Learn how a module corresponds to a namespace and be able to contrast the different ways to import module attributes.

Understand the Application

Your task this lab is to design, implement and test your own class called **Message** that models a basic email application.

Specifically you will write a program that will define a simple class that will model an e-mail message. You will then write a test driver program to validate your class email app.

You will define the **Message** class. The Message will include a sender, recipient and body. The sender is the originator of the email message. The recipient is the receiver of the message. The body contains the text of the message.

Provide the following methods for the Message class:

A **constructor** with parameters self, sender and recipient. Set values for sender and recipient. Set body as an empty string.

A method **append** with parameters self and line. line contains the line of text to add to the body of the email message. End each email message line with a newline character.

A method **to_string** that returns a string representation of the entire message. The entire message includes the sender, the recipient and the body of the message. Example: "From: Student One\nTo: Student Two\nLunch today?\n"

A helper method **str_ok()** that validates str parameters.

The Program Specification

The assignment is to first create a class called Message. Message will consist of three **instance attribute strings** as its basic data. It will also contain a few **instance methods** to support that data.

Once defined, you will use the Message class to **instantiate** a Message object(s) that can be used in your test driver **main** program.

Class Message

Instance Members:

- (string) **sender**
- (string) **recipient**
- (string) **body**

All legal strings should be of length ≤ 50 consisting of printable characters.

Instance Methods:

Constructor

Message() -- a parameterized constructor that initializes all class members.

def __init__(self, sender, recipient):-- a constructor that initializes all members according to the passed parameters. The constructor has to be sure each parameter **string** satisfies the class requirement for a member **string**. It does this, as shown in the lab forum discussion thread, by calling the mutators and taking possible action if a return value is false. If any passed parameter does not pass the test, a **default string** should be stored in that member.

Mutators

set() methods for the sender and recipient instance members. using the convention as follows:

set_sender(..), set_recipient(...).

append() method for the body instance member.

Mutators make use of the helper method **str_ok()** to validate parameter data.

When the set methods detect an invalid **string**, the mutator returns **False** and a new default String is stored in that member.

When the append method detects an invalid **string**, no action should be taken.

Helper Methods

string to_string() - a method that returns a **string** which contains all the information (three strings) of the **Message** object. This **string** can be in any format as long as it is understandable and clearly formatted.

def str_ok(self, the_str): -- a helper method that the mutators can use to determine whether a **string** is legal. This method returns **True** if the string's **length is <= MAX_LEN** (inclusive) and the string consists of printable characters. It returns **False**, otherwise.

Deliverables: Your source code solution module for the Message class *and* a test driver that will import the module. Have your class test driver contain a copy of the test run pasted into this file. Be sure to comment out your run so that your **.py** file will still run in the grader test bed.

Testing Specification

Input Error Checking: For simplicity a python test file is provided to begin testing your class. The filename is demo_message.py provided below.

Test Run Requirements: Extend the provided demo_message.py as your test run validator to both display your email message as well as to include demonstration that the set methods correctly reject invalid parameter data.

Here are some other *tips* and **requirements**:

1. Name the class **Message**.
2. Use a named constant to set the default string to the empty string.
3. Update the sender name to *your name* (i.e. name Ann should be changed to your first name). Update (i.e. customize) the wish list *items* to items of *your* choice. *If you wish*, update the recipient.
4. **Import** your Message class module into your test demo file.
5. Include the commented out copy of your test run in your demo file.

Here is a sample run:

...


From: **Ann**

To: Santa

For Christmas, I would like:

Video games

World peace

[demo_message.py \(https://smccd.instructure.com/courses/52459/files/9677573/download?wrap=1\)](https://smccd.instructure.com/courses/52459/files/9677573/download?wrap=1) 
(https://smccd.instructure.com/courses/52459/files/9677573/download?download_frd=1)

What to Turn In

1. Hand in only **one .py** class file.
2. Use this title format for your **.py** file : yournameLab4.py (*Note*: Use **YOUR** name).
3. Hand in only **one .py** test driver file.
4. Name the output file demo_yourname.py (Again, use **YOUR** name).

You will hand in **2** files: 1) your source file (which will contain the Message class) and 2) the demo test driver file (which will contain the commented out run of the output at your console).

Submitting multiple files to an assignment

Your lab assignment requires uploading more than one file; you should upload these **2** files as one submission. In this assignment you need to upload a **.py** file that contains your Message class and a **.py** file that contains your class test driver - both part of one assignment submission. To add these files, the **Add Another File button** is clicked to **upload the two files one by one**. **Check to make sure that both files uploaded okay**. When finished click **Submit Assignment**.

Submission Resources

For more information on how to submit your assignment, please visit:

- [How do I submit an online assignment? Canvas Student Guide \(https://community.canvaslms.com/docs/DOC-9539\)](https://community.canvaslms.com/docs/DOC-9539)
- [Assignments Overview Canvas Video Guide \(https://community.canvaslms.com/videos/1122-assignments-overview-students\)](https://community.canvaslms.com/videos/1122-assignments-overview-students)
- [Assignments Submissions Canvas Video Guide \(https://community.canvaslms.com/videos/1121-assignment-submissions-students\)](https://community.canvaslms.com/videos/1121-assignment-submissions-students)

Questions?

Feel free to [ask \(https://smccd.instructure.com/courses/52459/discussion_topics/823625\)](https://smccd.instructure.com/courses/52459/discussion_topics/823625) in the forum!

Lab 7 Rubric

Criteria	Ratings			Pts
On time submission Submitted on time 4 points One day late -2 points Two days late -4 points	4 pts Full Marks	2 pts Rating Description	0 pts No Marks	4 pts
Source Satisfies the source statements as specified in the assignment.	7 pts Full Marks		0 pts No Marks	7 pts
Class Test Driver - Includes commented out copy of test run. A test driver instantiates and demonstrates a Message object. A copy of the program run output is attached after the source code (enclosed within comment delimiters). The run matches the source code submitted.	7 pts Full Marks		0 pts No Marks	7 pts
Meets Coding Style Requirements Includes a program header. Is correctly formatted (i.e. follows code style rules).	2 pts Full Marks		0 pts No Marks	2 pts
Total Points: 20				