CS 8: Introduction to Computer Programming with Python
Summer 2017

Project 3

Assigned: Thursday, July 20 **Due:** Thursday, August 3 11:59 PM

Overview

In this project, you will create a simple mail merge program. This is a program that is used to customize form letters for multiple recipients at once. You will create a menu-driven program that allows the user to enter multiple recipients' information: their full name, short name (familiar name), and address.

The user will be able to write a letter template, stored as a plain text file, including several placeholders. By using the mail merge functionality of your program, the user should be able to make multiple copies of the letter template: one for each recipient. In each copy, the placeholders should be filled with the values associated with the intended recipient.

The set of placeholders is shown below.

Placeholder	**Substituted text**
`{shortname}`	Recipient's familiar name
`{fullname}`	Recipient's full name
`{address}`	Recipient's address

A sample letter template:

```
{fullname}
{address}

Dear {fullname},

Hi there, {shortname}! Hope all is well.

Have a good summer,

Bill
```

In this project, you will start by coding the basic menu and input validation. In successive activities, you will gradually add features.

You will need to show your work to your TA as you complete each activity; you will be graded on **both** the final code submission (60%) **and** the completion of your activities in recitation (40%).

You should try hard not to fall behind on the project. If you do not complete a session's activities during the session, you should complete them on your own before the next session and/or attend office hours for help. **No late assignments will be accepted.** This applies to both activities and final code submissions. By Thursday, August 3, you must:

- Show the result of each activity to the TA.
- Upload your code to Pitt Box, to the **provided** folder named `cs8-proj3-abc123`, where `abc123` is your Pitt username.

Note that you may submit to Box multiple times; you will be graded only on the last submission made before the deadline. You are encouraged to submit early and often, even if you are not finished, rather than risk getting a 0 by failing to submit on time.

Lab Session 1, Jul 20

Activity 1

In this activity, you will create the menu that is used to navigate the program.

Write a function called `menu()` that handles displaying the menu, prompting for input, and error-checking (via an input validation loop). Use all of the input-validation techniques we have learned to ensure that the program does not crash, and only a truly valid input is accepted. Once `menu()` has confirmed that the input is valid, it should return the user's choice.

Note that `main()` should allow the user to make one valid choice, but may need to display the menu multiple times if the user provides invalid input.

The menu should allow the user to apply a mail merge, list the current recipients, add a new recipient, delete a recipient, import a recipient list, export a recipient list, or quit. It should look something like this:

```
Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
```

You are welcome to use a different menu system as long as the required options are available, and your input validation ensures that only valid responses are accepted.

Once you have programmed the `menu()` function, create a `main()` function that calls this menu in a loop until the user chooses to quit. Test that the input validation works properly.

Each call to `menu()` should prompt the user until a valid input is given, and then return the choice. Thus, `main()` should store the result of this call to `menu()` so that it can determine which action the user wants to carry out.

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

Activity 2

In this activity, you will implement a simple recipient list as a list of strings.

1. In `main()`, add a local variable to store the recipients list. Initialize this to an empty list.

   Add code in `main()` to list the recipients if the user makes the corresponding menu selection. An example run is shown below. Since there are no recipients yet, the list is empty.

   ```
   Enter your choice:
   1) Apply a mail merge
   2) List recipients
   3) Add recipient
   4) Delete recipient
   5) Import recipient list
   6) Export recipient list
   7) Quit
   2

   Recipient list
   --------------
   --------------
   ```

2. Add code in `main()` to add a recipient if the user makes the corresponding menu selection. Prompt for the recipient's name, ensure that the recipient does not already exist, and add this name to the list. An example run is shown below.

   ```
   Enter your choice:
   1) Apply a mail merge
   2) List recipients
   3) Add recipient
   4) Delete recipient
   5) Import recipient list
   6) Export recipient list
   7) Quit
   3

   Name: Bill

   Enter your choice:
   1) Apply a mail merge
   2) List recipients
   3) Add recipient
   ```

```
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
3

Name: Jill

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
3

Name: Bill
Recipient exists
Name: Phil

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

Recipient list
--------------
Bill
Jill
Phil
--------------
```

3. Add code in `main()` to delete a recipient if the user makes the corresponding menu selection. Prompt for the recipient's name, ensure that the recipient exists, and remove this name from the list. Allow the user to type nothing (simply press enter) to cancel the removal. An example run is shown below.

```
Enter your choice:
```

```
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
4

Name (blank to cancel): Dill
No such recipient
Name (blank to cancel):

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
4

Name (blank to cancel): Bill

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

Recipient list
--------------
Jill
Phil
--------------
```

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

End Lab Session 1

Lab Session 2, Jul 27

Activity 3

In this activity, you will replace the single-string representation of recipients with an object-oriented representation.

1. Write a class named `Recipient` that represents a single recipient. Write the initializer `__init__(self, shortname)` that accepts the recipient's short name. Store the short name in a private data attribute, and initialize public data attributes `fullname` and `address` to empty strings. Add a public property (remember the `@property` notation) named `shortname` to access (but not change) this recipient's short name.

2. Write a function `input_address()` that prompts the user for a recipient's address. Since addresses can contain multiple lines, allow the user to continue typing until they have typed a blank line (i.e., pressed enter twice). Return the address as a single string (with newline characters to separate lines as needed). An example run is shown below.

   ```
   Address:
   123 Something Road
   Apt 1B
   Pittsburgh, PA
   ```

3. Replace the list of recipients with a dictionary. The key should be the recipient's short name (which must be unique), and the value should be a `Recipient` object. Modify the code to add a new recipient so that it prompts for a short name, full name, and address (the final one using `input_address()`). The short name should be provided to the initializer, while the full name and address can be assigned directly, since they are public. Update the list and removal code if necessary, to support a dictionary instead of a list. An example run is shown below.

   ```
   Enter your choice:
   1) Apply a mail merge
   2) List recipients
   3) Add recipient
   4) Delete recipient
   5) Import recipient list
   6) Export recipient list
   7) Quit
   3

   Short name: Bill
   Full name: William Garrison
   ```

```
Address:
987 Bucket Street
Campbell, PA

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
3

Short name: Bill
Recipient exists
Short name: Jill
Full name: Jillian Dillian
Address:
100 Jalopy Road
Pittsburgh, PA

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

Recipient list
--------------
Bill
Jill
--------------

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
```

```
7) Quit
4

Name (blank to cancel): Bill

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

Recipient list
--------------
Jill
--------------
```

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

Activity 4

In this activity, you will implement the mail merge functionality.

1. Add a new method to the `Recipient` class called `mailmerge(self, text)` that accepts a string parameter, and returns a copy of that string with each placeholder replaced with the appropriate value from the `Recipient` object.

2. Add code in `main()` to complete a mail merge if the user makes the corresponding menu selection. Prompt for the file name of the letter template, and read the contents of this file. The filename should end in the `.txt` extension.

   For each recipient in the dictionary, open a new file for writing, pass the template contents to the recipient's `mailmerge` method, and output the result to the new file. If the original file is `letter.txt`, then Bill's filled copy should be named `letter.Bill.txt`, Jill's should be `letter.Jill.txt`, and so on. Don't forget to catch any exceptions!

   An example run is shown below.

```
Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

Recipient list
--------------
Jill
Phil
Bill
--------------

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
1
```

```
Template filename: lettre.txt
Could not open file for reading

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
1

Template filename: letter.txt
Mail merge completed
```

An example filled letter, `letter.Jill.txt`, is shown below.

```
Jillian Dillian
100 Jalopy Road
Pittsburgh, PA

Dear Jillian Dillian,

Hi there, Jill! Hope all is well.

Have a good summer,

Bill
```

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

Activity 5

In this activity, you will implement import and export of the recipient list using pickling.

1. Add code in `main()` to export the recipients list if the user makes the corresponding menu selection. Pickle the dictionary to `recipients.bin`. An example run is shown below.

```
Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

Recipient list
--------------
Phil
Jill
Bill
--------------

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
6

Recipients exported to recipients.bin
```

2. Add code in `main()` to import the recipients list if the user makes the corresponding menu selection. Un-pickle the dictionary from `recipients.bin`. Ensure that a mail merge works properly after an import, even from a fresh launch of your program. An example run is shown below.

```
Enter your choice:
1) Apply a mail merge
```

```
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

Recipient list
--------------
--------------

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
5

Recipients imported from recipients.bin

Enter your choice:
1) Apply a mail merge
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

Recipient list
--------------
Bill
Jill
Phil
--------------
```

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

End Lab Session 2