CS 0011: Placement Project
2019

Overview

In this project, you will create a simple program for creating *form letters*, letters written 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 *substitution* 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 you must support is shown below.

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

A sample letter template might look like this:

```
{fullname}
{address}

Dear {fullname},

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

Have a good summer,

Bill
```

In this project, you should start by coding the basic menu and input validation, gradually adding additional features.

Specifications

- You should have a function called `menu()` that handles displaying the menu, prompting for input, and error-checking (via an input validation loop). `main()` should allow the user to make only one valid choice, but may need to display the menu multiple times if the user provides invalid input. Once `menu()` has confirmed that the input is valid, it should return the user's choice as an int.

  The menu should allow the user to apply the substitutions to a template, 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) Create letters from template
  2) List recipients
  3) Add recipient
  4) Delete recipient
  5) Import recipient list
  6) Export recipient list
  7) Quit
  ```

  Additionally, you should have a `main()` function that calls this menu in a loop until the user chooses to quit. Each call to `menu()` should prompt the user until a valid input is given, and then return the choice.

- You will need to maintain a collection of recipients for the template. This collection should be stored as a dictionary. The keys of this dictionary should be recipient's short names (which must be unique), and the values should be `Recipient` objects. Initially, this list should be empty. When the user chooses to add a recipient, you should insert a recipient object into the list. You will need to write a class named `Recipient` that has the following:

  - An initializer `__init__(self, shortname)` that accepts the recipient's short name, stores the short name in a private data attribute, and initializes the public data attributes `fullname` and `address` to empty strings.
  - Add a public property (using `@property` notation) named `shortname` to access (but not change!) this recipient's short name.
  - A new method called `substitute(self, text)` that accepts a string parameter, and returns a copy of that string with each placeholder substituted with the appropriate value from the `Recipient` object.

  You should further write a function `input_address()` (outside of the `Recipient` class) 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).

When the user adds a new recipient, your `main()` function should prompt for a short name, full name, and address (using `input_address()` for the address). The short name should be provided to the initializer, while the full name and address can be assigned directly, since those attributes are public.

- When the user chooses to "Create letters from template", your `main()` function should perform a substitution for each recipient. Prompt for the file name of the letter template, and read the contents of this file. The filename should end in the `.txt` extension. Then, For each recipient in the dictionary, open a new file for writing, pass the template contents to the recipient's `substitute` 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 to prevent your program from crashing!
- When the user chooses to "Export recipient list", your `main()` fucntion should pickle the dictionary to `recipients.bin`.
- When the user chooses to "Import recipient list", your `main()` fucntion should unpickle the dictionary from `recipients.bin`. Be sure to test that substitution works properly after an import, even from a fresh launch of your program.

An example run of the expected program is shown below:

```
Enter your choice:
1) Create letters from template
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) Create letters from template
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) Create letters from template
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) Create letters from template
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) Create letters from template
2) List recipients
3) Add recipient
4) Delete recipient
```

```
5) Import recipient list
6) Export recipient list
7) Quit
2

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

Enter your choice:
1) Create letters from template
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

Enter your choice:
1) Create letters from template
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
7

Goodbye!
```

A second example run is shown here:

```
Enter your choice:
1) Create letters from template
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) Create letters from template
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) Create letters from template
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
2

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

Enter your choice:
1) Create letters from template
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) Create letters from template
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
1

Template filename: letter.txt
Substitution completed

Enter your choice:
1) Create letters from template
2) List recipients
3) Add recipient
4) Delete recipient
5) Import recipient list
6) Export recipient list
7) Quit
7

Goodbye!
```

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
```