HÖGSKOLAN
DALARNA

Python Programming Assignment

Notes Handler

Project Report

Saumya Gupta

17 March 2021

Table of Contents

Introduction

This report explains in detail the solution developed for a python programming assignment, solely by the author, as part of the '*Python- and R-programming*' course held at Dalarna University for the master's in the data science program. The assignment, a mini project to be precise, asks to create an object-oriented program, a notes handler, as an individual effort towards learning and strengthening the concepts of python programming with an object-oriented approach. It also tests our capability as a program designer in trying to work with the problem domain, identifying potential classes and giving the best user experience possible.

Notes Handler is a program that on execution helps the user keep track of his/her notes over time. It facilitates basic CRUD (create, read, update, delete) operations of persistent storage on notes. Apart from these basic operations, users can also perform complex tasks such as restoring notes from the previous session, getting completion statistics on created notes, to list a few. The program is user friendly with its elaborative menu system. It tries its best to guide the user at every step as to what has been done and what must be done, hence is an easy-to-use tool for a new user.

Following object-oriented principles, the program consists of classes and the main program, explained in further details in sections to come. First, classes interact with each other to complete the tasks requested by the user. Second, while doing all that it practices input validation wherever user input enters the program. Third, being a mini utility, it uses a sequential access text file for storage. Lastly, it shows requested data in tabulated forms for a good UX.

Designing and developing such a solution was a wonderful learning experience, especially where the problem description was inexplicit enough to give room to numerous possibilities in implementation. I explored various ways of implementing the same feature, working as a designer, a developer, and a tester thought over numerous user scenarios which helped me refactor my code towards a better result.

Description

As mentioned before, the solution follows an object-oriented approach and hence consists of two classes, **Notes** and **userInterface** and the **main program**. The program can be viewed as involving 3-levels of processing. The main program is invoked as soon as the program is executed (See User Guide to know how to run the program). After performing its tasks, it passes control to the userInterface class, which then is responsible for interacting with the Note class to fulfil the user request. Figure 1 shows the UML diagram for the classes:

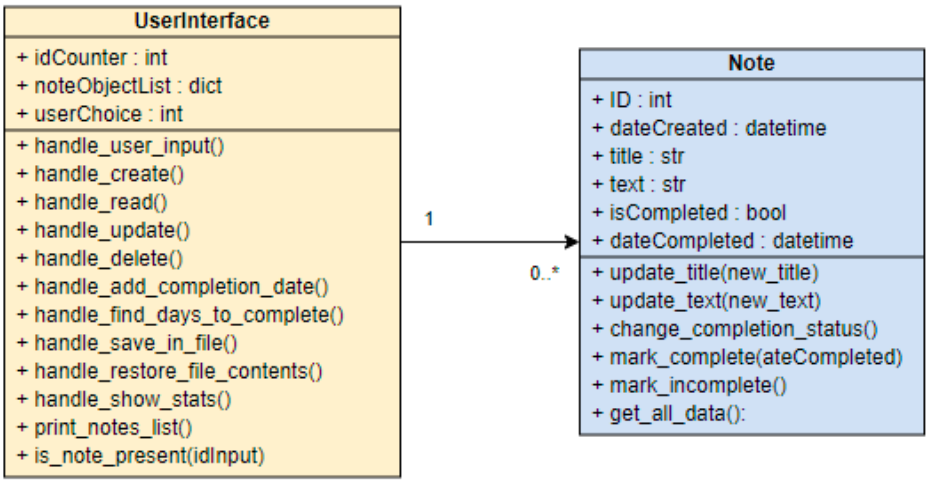

Figure 1. UML diagram showing the class design and relationships.

Following are the details of classes used and the functions they interact with:

- Note class: This class represents a single note. Every note created by the user in the program is an object, i.e., an instance of this class. In the code, it is created using an object of userInterface class and after creation, is saved in a userInterface class attribute.
  This class is responsible for knowing note metadata and performing operations on the same including updating note data, showing note data, and changing note completion status, to name a few. Details of class attributes and methods are given below:

  Attributes

  ID: (int) Unique identification number of the note

  dateCreated: (datetime) Creation date of the note

  title: (str) Title of the note

  text: (str) Text/body of the note

  isCompleted: (bool) True if the note is completed, otherwise False

  dateCompleted: (datetime) Completion date of the note

  Methods

  update_title(new_title): Updates the title of the note

  update_text(new_text): Updates the text/body of the note

  change_completion_status(): Toggles the completion status of the note

  mark_complete(dateCompleted): Marks the note complete

  mark_incomplete(): Marks the note incomplete

  get_all_data(): Returns a dictionary of the note's data

- userInterface class: This class represents the user interface with its menu system. The main program is responsible for creating an object of this class for a session. After a session gets over and the user is offered the same menu system again in the next session, the main program creates another object with the same object name. (There could be other ways of implementing class object initialisation, but given the time frame, proceeding with this

method seemed practical). The objects of this class are not stored anywhere unlike the Note class' objects, rather overwritten every session.
This class is responsible for interacting with the user after the user responds to the primary menu item. To keep it short, it knows what action must be performed to fulfil the user's specific request and prompts the user in case more information is needed. Details of class attributes and methods are given below:

Class Attributes

idCounter: (int) Represents the note IDs; increments on note creation

noteObjectList: (dict) Dictionary with the note IDs as keys and the corresponding Note class' objects as values

Instance Attributes

userChoice: (int) Input entered by the user, in response to the primary menu item

Methods

handle_user_input(): Decides which handler to call based on the '*userChoice*' input

handle_create(): Creates the note object

handle_read(): Displays specific note object data

handle_update(): Updates specific note object

handle_delete(): Deletes specific note object

handle_add_completion_date(): Adds/updates the completion date to/of note object

handle_find_days_to_complete(): Displays the number of days it took to complete the note

handle_save_in_file(): Saves all the current note objects into a file

handle_restore_file_contents(): Restores all the notes from saved file; DOES NOT overwrite already existing notes in the current session

handle_show_stats(): Displays the notes statistics focussing note completion

print_notes_list(): Displays the list of notes

is_note_present(idInput): Checks if the concerned note is present in the '*noteObjectList*' attribute

The main program is responsible for displaying the primary menu item, taking input for it from the user, validating it and creating a userInterface instance to handle the user's choice. Once the task is handled, control returns to the main program, where on receiving another user choice, it repeats the process of overwriting the object variable to reference a different object of the same class. This keeps going until the user chooses to exit.

User Guide

Running Notes Handler:

- To run Notes Handler, the user needs to have the python interpreter installed in their systems. The program has been built using Python 3.9.1 version. It is advisable to install or switch to the same version to avoid any disparity in program execution. Python can be installed from this link: https://www.python.org/downloads/.
- The program is using certain packages from PyPI to ensure tabular output and pretty display. User must ensure that all the packages mentioned in the '*requirements.txt*' file are available within the environment he/she would like to work in. If not then the user can install it using pip or conda, depending on whatever he/she is comfortable with. If the user wants to go with pip, he/she can install it following the directions given in the link, if not already installed: https://packaging.python.org/tutorials/installing-packages/#requirements-for-installing-packages
- Change to the directory where program files are present. There should be 4 files available: '*Main.py'*, '*Notes.py*', '*userInterface.py*' and '*requirements.txt*' to install required packages. Once there, open the terminal and use the command below to install all packages in one go:

  pip install -r requirements.txt

  At the end of the day, the user needs to make sure that all packages are installed in the environment from where he/she plans to execute the program.
- To run the program, use the following command:

  python Main.py

  Notes Handler is running now, and the user should see the primary menu item, waiting for the user response.

Using Notes Handler to handle your notes:

Once the program is up and running, the user can work with the various options available. The program is self-explanatory and guides the user at every step. In any case following content helps user tackle confusion:

Q – How to add a new note?

1. Enter '1' from the primary menu.
2. Enter note text, title, and completion status as and when prompted. Keep note of the short text available in '*()*'. It guides the user on what is accepted and what is not.
3. Empty strings or invalid inputs for completion status, result in error traps and keep prompting the user to correct their input.
4. If every input is in line, a note is created.

Q – How to read a note?

1. Enter '2' from the primary menu.
2. If no notes have been created user is asked to either create a new note or restore notes from a previous session.
3. If notes are available in the system, they are printed to give the user the information about what notes can be chosen to do the requested operation on.

4. In the latter case, enter the ID of the note to be read.
5. If a fresh note ID is entered, the program exits the operation for a fresh session and gives a chance to the user to enter a known note ID next time.
6. If the entered note ID is known to the program, the user is shown the details of the note requested.

Q – How to update a note?

1. Enter '3' from the primary menu.
2. If no notes have been created user is asked to either create a new note or restore notes from a previous session.
3. If notes are available in the system, they are printed to give the user the information about what notes can be chosen to do the requested operation on.
4. In the latter case, enter the ID of the note to be updated.
5. If a fresh note ID is entered, the program exits the operation for a fresh session and gives a chance to the user to enter a known note ID next time.
6. If the entered note ID is known to the program, the user is shown the details of the note requested and prompted with the second menu item asking what note attribute is to be updated. At this time user can exit if he changes his mind about updating.
7. On selecting the attribute to be updated, the user is prompted to enter the new values.
8. Empty strings result in error traps and keep prompting the user to correct their input.
9. If every input is in line, the note is updated.

Q – How to delete a note?

1. Enter '4' from the primary menu.
2. If no notes have been created user is asked to either create a new note or restore notes from a previous session.
3. If notes are available in the system, they are printed to give the user the information about what notes can be chosen to do the requested operation on.
4. In the latter case, enter the ID of the note to be deleted.
5. If a fresh note ID is entered, the program exits the operation for a fresh session and gives a chance to the user to enter a known note ID next time.
6. If entered note ID is known to the program, the note is deleted.

Q – How to add a completion date manually?

1. Enter '5' from the primary menu.
2. If no notes have been created user is asked to either create a new note or restore notes from a previous session.
3. If notes are available in the system, separate lists for completed and non-completed notes are printed to give the user the information about what notes can be chosen to do the requested operation on.
4. In the latter case, enter the ID of the note to add a completion date.
5. If a fresh note ID is entered, the program exits the operation for a fresh session and gives a chance to the user to enter a known note ID next time.
6. If a completed note ID is entered, the user is prompted to enter a completion date and the completion date for the note (previously set to creation timestamp) will be updated to the entered completion date.
7. If a non-completed note ID is entered, the user is prompted to enter a completion date and the completion date for note (previously set to None) will be updated to the entered completion date. The corresponding note will now enter the completed notes list.

8. The completion date must be entered in the format specified. Empty strings or invalid inputs result in the program going to the main menu and the current operation terminates.

Q – How to find the number of days it took to complete a specific note?

1. Enter '6' from the primary menu.
2. If no notes have been created user is asked to either create a new note or restore notes from a previous session.
3. If notes are available in the system, they are printed to give the user the information about what notes can be chosen to do the requested operation on.
4. In the latter case, enter the ID of the note to find this information.
5. If a fresh note ID is entered, the program exits the operation for a fresh session and gives a chance to the user to enter a known note ID next time.
6. If the entered note ID is known to the program, the requested info is displayed.
7. If the creation and completion timestamp belong to the same date, irrespective of the difference in the exact hours, seconds or the like, the result is 0 days.

Q – How to save all notes in the program to a file?

Warning – Always restore before saving.

1. Enter '7' from the primary menu.
2. If no notes have been created user is asked to either create a new note or restore notes from a previous session because there is nothing to save!
3. If notes are present in the program, and no '*Notes.txt*' file has been created before, all notes in the program are saved to the file with the same name.
4. If there are notes already saved in the '*Notes.txt*' file that are not known to the program, for example, notes from the previous session or notes from the same session, i.e., saved and then deleted from the program for whatever reason, the user is prompted to rethink his decision since saving always overwrites the output file.
5. In the above case, if the user wants to go back to the main menu and restore, he/she should enter '1', else '0'. Entering '0' will overwrite the file and the user will lose the notes in the file that are unknown to the program.

Q – How to restore notes saved in a file to the program?

Note - Restoration from the file only brings in notes unknown to the current session of the program, if there are same note ID's also present in the current session, they are not overwritten.

1. Enter '8' from the primary menu.
2. If there is a '*Notes.txt*' file present in the directory, all the notes from there are restored to the program. This option is useful if you have notes saved from the previous session and want to work with them again.
3. If no '*Notes.txt*' file is present, it asks to create a new note or save notes from the current session to file first.

Q – How to find how many notes have been created and how many of them are completed?

1. Enter '9' from the primary menu.
2. The user is displayed with the information requested.
3. If no notes have been created, the percentage of notes in a non-completed state is shown with '*Not applicable*' text, because for obvious reasons.

Q – How can the user exit the program?

1. Enter ‘10’ from the primary menu.
2. The program terminates.

Resources

1. Gaddis, T. (2019). *Starting Out with Python* (4th ed.). London: Pearson Education Limited.
2. McKinney, W., & others. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51–56).
3. https://pyyaml.org/wiki/PyYAMLDocumentation
4. https://pypi.org/project/tabulate/
5. https://docs.python.org/3/library/datetime.html
6. https://docs.python.org/3/library/os.html
7. https://docs.python.org/3/library/sys.html