Hedy Khalatbari

August 26, 2021

Foundations of Programming: Python

Final

GitHub URL: https://github.com/Seattle15/Intro\_to\_Prog\_Python\_FinalHK

Programming with Python: Final

# Step 1. Create an Application

Program an application that **tracks the number of hours an employee works each day on one or more projects**. The application **must read, add, and delete data** from the provided EmployeeProjectHours.csv file.

## Check List

* Create a new **sub-folder called FinalYourNameHere**
* Create a **new project** in PyCharm that uses your folder
* Create script and add pseudo-code
* Include a script header
* Create IO and Processing classes
* Create and test the functions
* Include function headers
* Put the functions in the classes and test them again as an integrated unit
* Add comments to your code to make it clearer to other developers
* Update the main body of the script to make using the application looks and feel more professional, then test it in Pycharm and the OS console.
* I systematically, and at times iteratively, went through the checklist and accomplished each step
* The application is comprised of three classes and the main script that have been organized based on the principle of separation of concerns:

1. **EmployeeHours class**: an object instance of this class corresponds to one entry row in the csv file. I defined a constructor as well as getters and setters for the five columns in the csv file (I added an EntryNum column that I use when deleting an entry); the five variables are EntryNum, EmployeeName, ProjectName, FullDate, and HoursWorked. The setters are used to validate (or reject) the values entered

* although I have tried to enforce some valid entries in the data entries, I have not been overzealous in parsing out why exactly an entry is rejected. When entries are rejected, the user gets a generic message

1. **Processor class**: the static methods in this class read and write to a csv file (using DictReader and DictWriter methods), add and delete entries to the list of dictionaries, show the list of entries, ‘show and add to the list of projects’, and reload data from file

* the application does not account for upper and lower case in project names if the spelling is consistent; project names cannot be numeric
* if os.path.exists(file\_name): --- is used to ascertain a file exists before attempting to read it
* if os.path.exists(file\_name): --- is also used to ascertain that if a file does not exist and is being created the headers are written to the cvs file using the writeheader method
* I also tried running the script without an already created csv file and ensured that the script could handle making a file from scratch and reading from an empty file without raising errors
* a **counter** is used to keep track of row entries (when initially loaded or when added by user); this is saved as the EntryNum value and is used to identify the entry the user wants to remove. Note that the EntryNum’s are only refreshed (i.e., set to sequential numbers starting from one) once the data is saved and reloaded from the csv file
* a **status** string is used to keep track of what methods are doing and providing feedbacks to the user
* **sort** – the application sorts the entries based on ProjectName when reading and writing the file as well as when it prints the list of entreis
* **renumbering -** the application renumbers the EntryNum of the sorted list when reading and writing the file

1. **IO class**: this class is an expansion of the IO class we first encountered in Assignment 6. The menu options are expanded to include a 6th option for showing the list of current projects and adding to it (Figure 1).

The static methods include:

* print\_header: prints a short introduction with instructions for the user (Figure 2)
* print\_menu\_options
* input\_menu\_choice
* print\_current\_Entries\_in\_list: prints all the entries sorted by ProjectName
* print\_current\_Projects\_in\_list: prints all project names (no duplicates) sorted alphabetically
* input\_yes\_no\_choice
* input\_press\_to\_continue
* input\_new\_entry: all data for an object instance are entered using this method and validated with the getter properties of the class
* input\_entry\_to\_remove
* input\_new\_project: to maintain consistent spelling a list of projects keeps track of the projects the employees are working on. Before a new project can be entered as an EmployeeHours object instance the projects has to be added to the projects list

1. **Main script**

* starts by printing the header, loading data from the EmployeeProjectHours.csv, and printing a list of the entries (sorted by Project Name) and a list of the projects currently in the csv list
* it then prints the menu options and asks the user to choose an option
* any new entries are initially added to the list of entries and only saved to the csv file once the user chooses that option
* note that while the user can add projects to the project list, they cannot delete projects from that list. This was an intentional choice due to the logic of the application
* a list of projects is also saved in lower case letters – this is to verify the validity of user entries in a letter case agnostic manner. That is, the application will treat ‘ETL’ and ‘etl’ the same way

Text

Description automatically generated

**Figure 1.** Menu of options for the application

Text

Description automatically generated

**Figure 2.** Introduction and instructions that are printed when the application initially runs

A picture containing text

Description automatically generated

**Figure 3.** Script running in PyCharm

Text

Description automatically generated

**Figure 4.** Script running in Command console. I intentionally did not provide the csv file in the same location to illustrate that the script can handle running when no csv file exists a priori

# Step 2. Document your Application

The ProjectTracker manual is submitted both here and as a separate pdf document.

# The ‘Project Tracker’ Manual

**Overview of the ‘Project Tracker’**

**What does it do? This application will allow you to track the time your team members spend on different projects**

**How is the data saved? The data is saved as a CSV file**

**What data is saved? Employee Name, Project Name, Date, and Hours Worked on project**

**How do I enter the data? The data is entered via the user input interface**

**Do I have to create the CSV file? The application will create the file for you**

**Can I change or delete entries? Entries may be deleted and re-entered with the appropriate changes**

**Step by step guide to using the application**

1. **Once you run the application the initial page will give you simple instructions**

**Text

Description automatically generated with medium confidence**

1. **The application will then print the following messages when you first run it. These inform you that a file does not currently exist and there are no employee-hour entries or projects in the lists**

**Text, letter

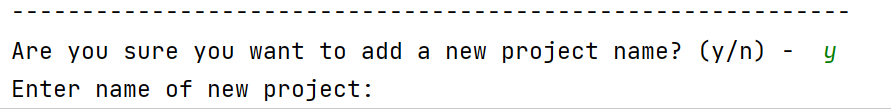
Description automatically generated**

1. **It will then print the menu of options and guide you to enter a project name**

Text

Description automatically generated

1. **Once you enter ‘y’ you can enter the name of your project**



1. **Once you enter the project name, you will be able to start entering employee-hours in the Project Tracker by choosing option ‘1’**

A picture containing text, indoor

Description automatically generated

Text

Description automatically generated

1. **You will see inputs for entering employee name, project name, date (in the correct format) and number of hours worked on the project**

Graphical user interface, text, application

Description automatically generated

1. **If the entries are in an incorrect format, you will see the following message**

**Text

Description automatically generated**

**In this example the project name was not in the project list; the employee’s last name was entered as digits, and the date was formatted incorrectly. Therefore, the entry was rejected**

1. **Menu of options: allows you to add or delete an entry**

**save data to a comma separated values (CSV ) file**

**read data from a CSV file**

**show list of all entries**

**show and add to project list**

**EXIT**

A picture containing text, indoor

Description automatically generated

1. **Once you have added the entries choose option 3 to save data to file.**

**The next time you start the application will automatically read the entries from the file and sort the entries based on the project names**

A close-up of a document

Description automatically generated with low confidence

**… and the print a project list of your current projects**

Text

Description automatically generated with medium confidence

1. **Option 2 allows you to delete an existing entry**

Text

Description automatically generated

1. **Option 4 allows you to reload data from the file. However, any entries since the last file save will be lost**

Text

Description automatically generated

1. **The entries are saved as a CSV file, that you can share with you colleagues and management**

**Table, Excel

Description automatically generated**

**Summary of feature of the ‘Project Tracker’**

**Track: Employee Name, Project Name, Date, and Hours Worked on project**

**Add and delete entries: using a user-friendly interface**

**Save entries as: a CSV file**

**Sort entries: based on alphabetical order of project names**

**We look forward to your feedbacks on how we can better fulfill**

**your requirements. Please visit us at ProjectTracker.org**

# Step 3. Write a Lessons Learned Document

My first exposure to a programming language was Visual Basics which I dabbled in at a beginner level twenty years ago. In 2015, I started learning more about data visualization with Tableau, which lead me to also delve a bit into R at that time. However, I got sidetracked with responsibilities and deliverables at work and there was a hiatus till the ‘COVID era’ when I decided to resume my quest in data sciences.

In March 2020 with the advent of COVID-19 our daily activities were disrupted. I lived and breathed either at work or in my living room at home, and I needed a hobby or activity which I could pursue from the coach in that living room. Preferably one that I enjoyed and found challenging but could also apply in my daily work in the future.

I work in healthcare as a pediatric radiologist – I look at images of the human body all day long, detecting normal and abnormal structures, and forming an interpretation of what I see for other providers. My specialty is very dependent on computers and applications. An application may be used to transfer multiple blood velocity measurements the sonographer has made to the report the radiologist generates or alternatively provide templates for a radiology report. How well those applications run has a huge impact on how efficiently I can get my work done – and they do not always run smoothly.

In addition, with the more recent renewed interest in machine learning (including its applications to imaging) there has been a wealth of publications on this subject in radiology. I wanted to contribute to that literature and to do so I need to learn machine learning – which may be an unrealistic goal given all my other responsibilities, but in the new world of COVID-19 I had the time and the will to take the first steps in this journey.

University of Washington offers multiple, online educational opportunities as ‘Professional and Continuing Education’ courses or certificate programs. To be eligible for the ‘Machine learning’ certificate program I had to fulfill several prerequisites (such as Python Programming and Data Sciences). ‘Foundations of Programming: Python’ was the first prerequisite course that I took in this 3-year long journey.

When I started learning Python my original thought was that this is the only programming language I will ever learn and then I will delve into data sciences and machine learning. However, the way this course was structured and taught made me curious about programming languages and reinforced that once I learn to think as a programmer learning a second or third programming language will be much easier.

I also discovered that I enjoy programming because of its inherent logic. In contradistinction to medicine in programming there is always a straightforward and logical way to do things (i.e., write script). And you can make mistakes – the script will fail to run and give you an error message unless it is a logical error – and correct those mistakes as you run and test your scripts. These ‘practice runs’ are fun and getting the script to work with efficiently written code is fulfilling!

I am a life-long learner and applied my experiences and knowledge in learning in the medical field to learning programming, including reading extensively and taking notes. I diligently spent 15-plus hours each week reading and understanding the materials offered in each module. I firmly believe that in life there are no short cuts and that you reap what you sow – and it all starts with the basics and having a solid foundation.

I think this course has laid the grounds for me to have a solid foundation in programming on which to build. I will follow this course, with the Python Programming certificate course and then the Data Sciences course. I will probably take detours into other programming languages and learn how to develop interactive web pages and applications.

I enjoyed listening to Mr. Root’s pre-recorded modules (and pausing them to dig deeper into a subject), Ms. McKeever’s detailed media feedbacks to our assignments, and sharing scripts and feedbacks with my fellow classmates. Thank you all!

# Step 4. Post the ‘Lessons Learned’ document to the Discussion Board

I will post this document to the discussion board once it opens for the finals

# Step 5. Submit your Work

I submitted my assignmet document and Python script on Canvas.

As the corresponding figure had mentioned posting it on GitHub (although not mentioned in the body of the assignment) I did that as well.

GitHub URL: https://github.com/Seattle15/Intro\_to\_Prog\_Python\_FinalHK