**Data Encoding and Processing**

<https://docs.google.com/document/d/1cTlgqNeDx1vXh3CMJRMNKvGCbpMxmcHbPQ5JZiH21fI/edit?usp=sharing>

<https://github.com/Deepanshu-TTN/bootcamp-git/tree/master/week-5/Data%20Encoding%20and%20Processing>

Refer to the pages below for github for the corresponding assignment, since this is a joined one

Q1)What is StringIO library for? explain with an example

StringIO module in python is used to create a file-like object. This object can be used as input and output to most of the methods which expect a file object like read() write() readlines() etc. StringIO object is initialized with a string literal passed in. If no string is passed the StringIO will start empty. In both cases, the initial cursor on the file starts at zero.

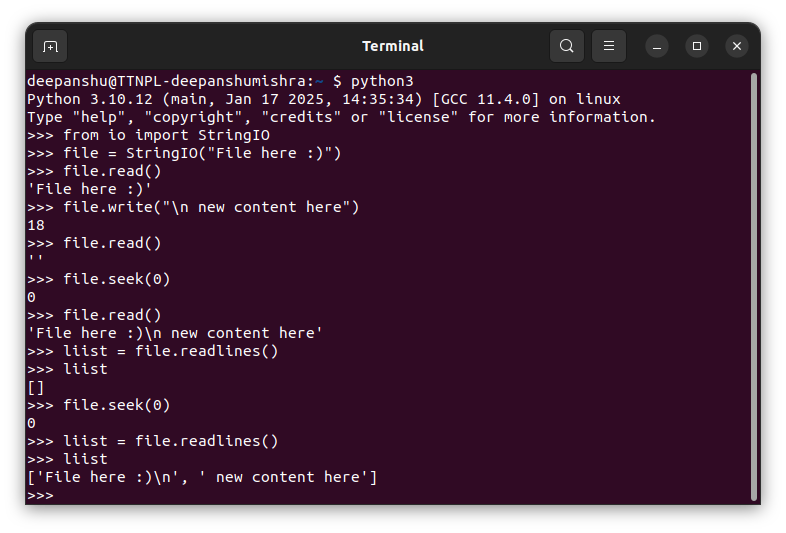
Basic use case for this module is to emulate a file, or we can simply compress a string to a gzip file without creating a file in the process or transferring logs of an application over the network again without touching the filesystem, it can be used in functions that require a file but we need to pass in a string.

We emulate a file using the StringIO object with some starter string.

file.read() reads the file like it does with a real file

When we use the write() method we get back a seek value which is at the end of the file

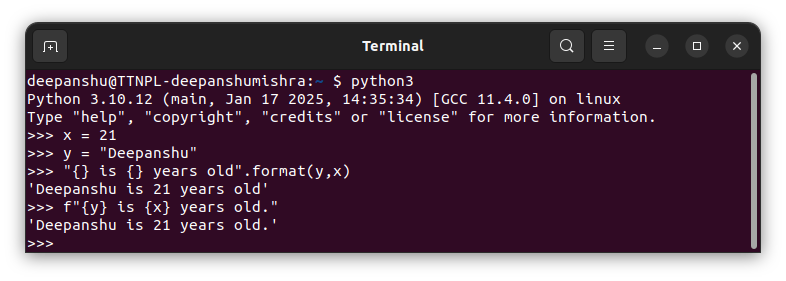
We seek back to ‘0’ or start of the ‘file’ and read, we can see the complete file content



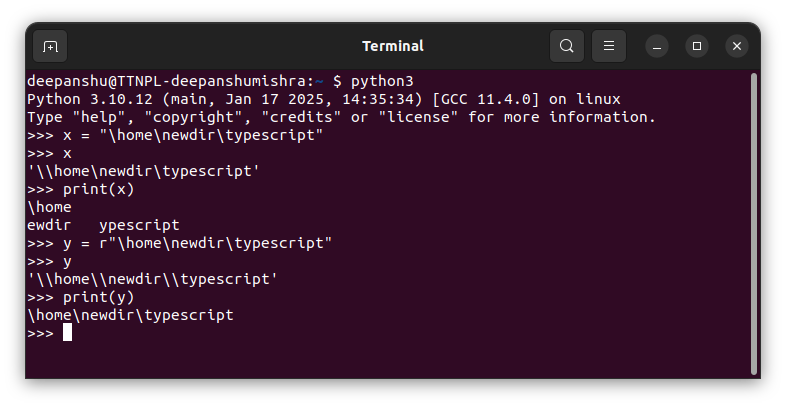
Q2)What is u"", r"" and "", f"" string notations in Python. Give example?

In python string literals can be prefixed with certain characters to indicate their type or behaviour.

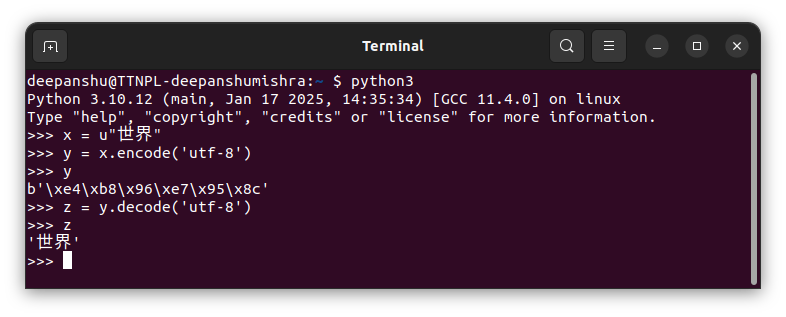
f’’or f-string: denotes a formatted string literal. It allows you to embed expressions inside string literals, using curly braces {}, this is an advancement to the previously used string.format() method in which you pass expressions within the parentheses



r’’ or raw-string literals: in a raw string, backslashes are treated as literal characters and do not escape characters like \n or \t in a usual string literal. This is useful for regular expressions or file paths. In the following example notice what happens when i print x and y:



u’’: denotes a unicode string. Python 2 had all strings as byte strings by default, and you needed to use the u prefix to create a Unicode string which could include characters from all the languages and symbol sets. In Python 3, all strings are Unicode by default, so the u prefix is not necessary, but it is still valid. Rather, we can use b’’ to create a byte-string and use the encode and decode functions to convert between unicode and byte strings

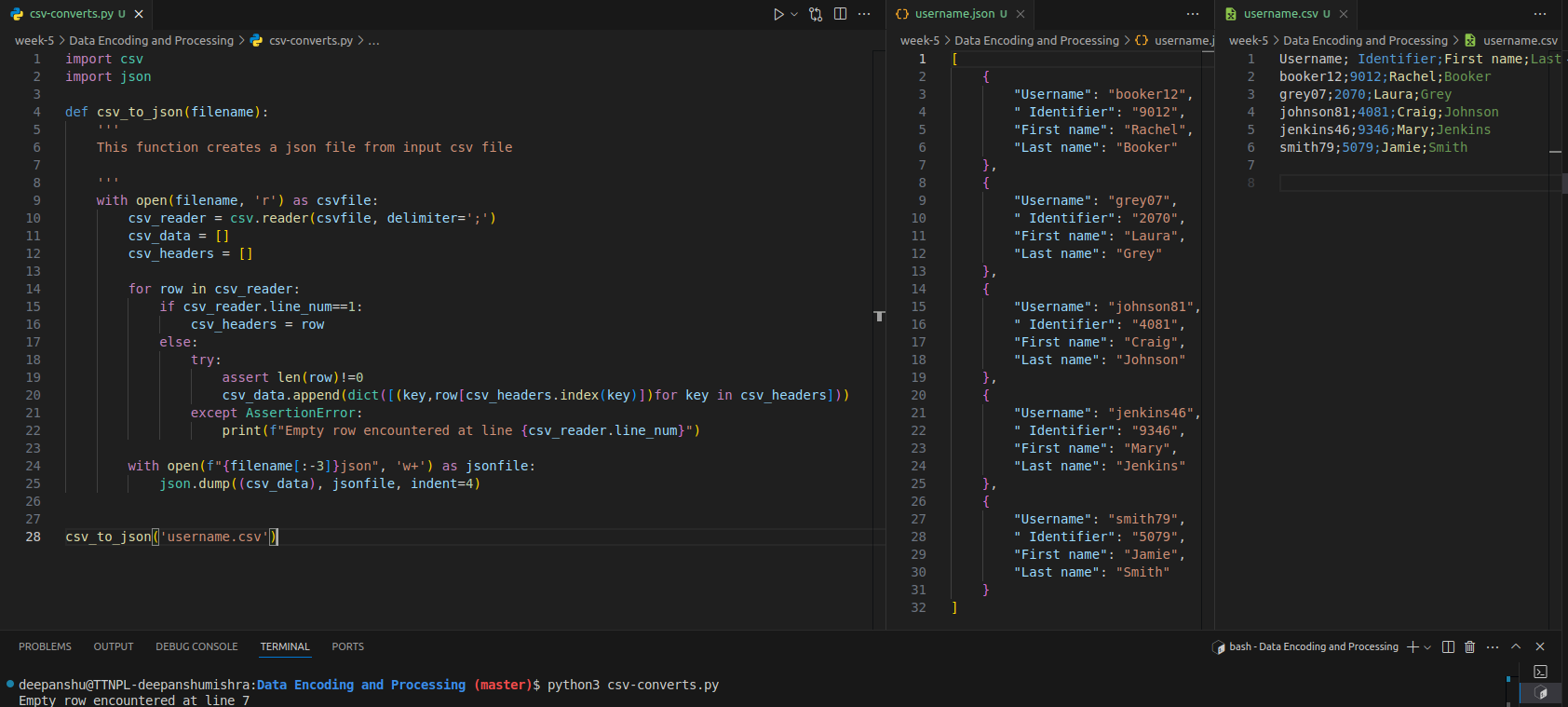


Q3) Part-1: Write a simple script to tranlate CSV file into JSON file

input: example CSV file

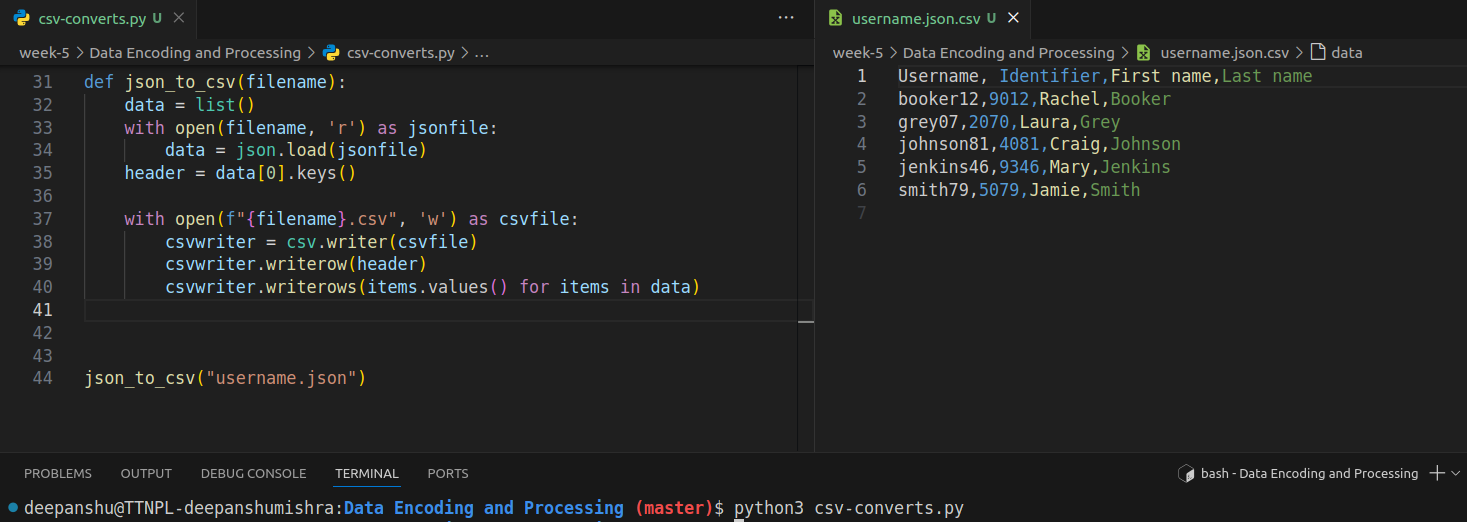
output: .json file - each line is valid JSON equvilent to a row in CSV file

In the following code username.csv is taken as the argument to the csv\_to\_json() function and the code inserts row data into the csv\_data variable and then dumps this to a new username.json file. The input csv and output json can be seen on the right hand side and the terminal gets prompted for every empty line encountered like in line 7 in username.csv.

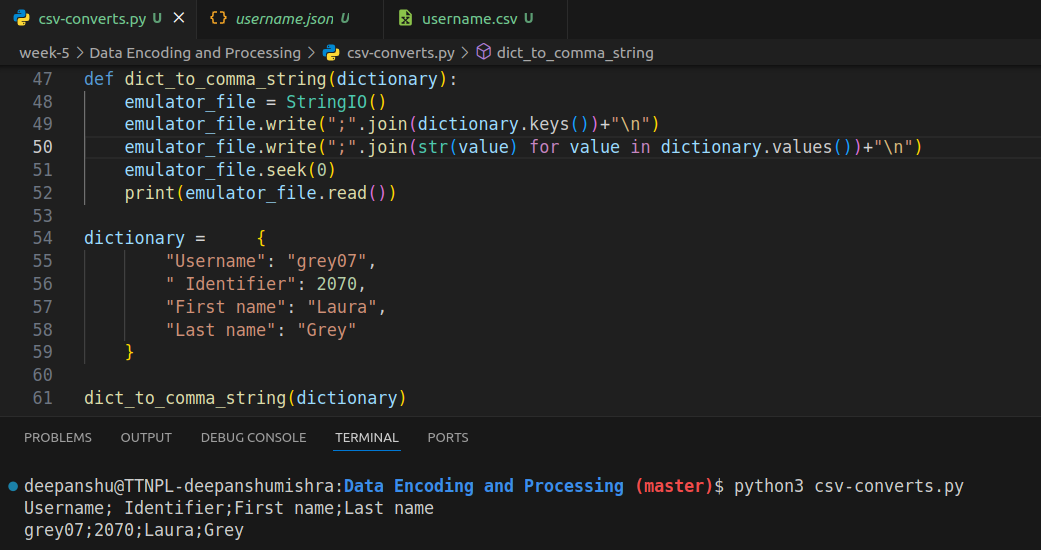


Part-2: Write a script to tranlate a simple JSON file (white each line is a valid JSON) into .CSV file Assume that JSONs are not nested

The code on left hand side takes the json generated earlier and writes the data into a username.json.csv file as seen on the right hand side window.



Part-3: Write a simple function using StringIO which can transalte a "dict" into a comma-seprated string



Q4)In Python, what is `bytes` and `bytearray` ? What is use of both and when to use both explain with a use-case code example.

How to covert a string literal to bytes and and bytes to string

* Bytes are immutable ie once a bytes object is created, it cannot be modified.and it is typically used when you want to represent binary data that should not change, such as reading data from a file or receiving data over a network.
* Bytearray on the other hand can be modified after it is created which can be useful when you need to manipulate binary data, such as modifying specific bytes or appending new bytes.

**Regular Expression and Text Proccessing**

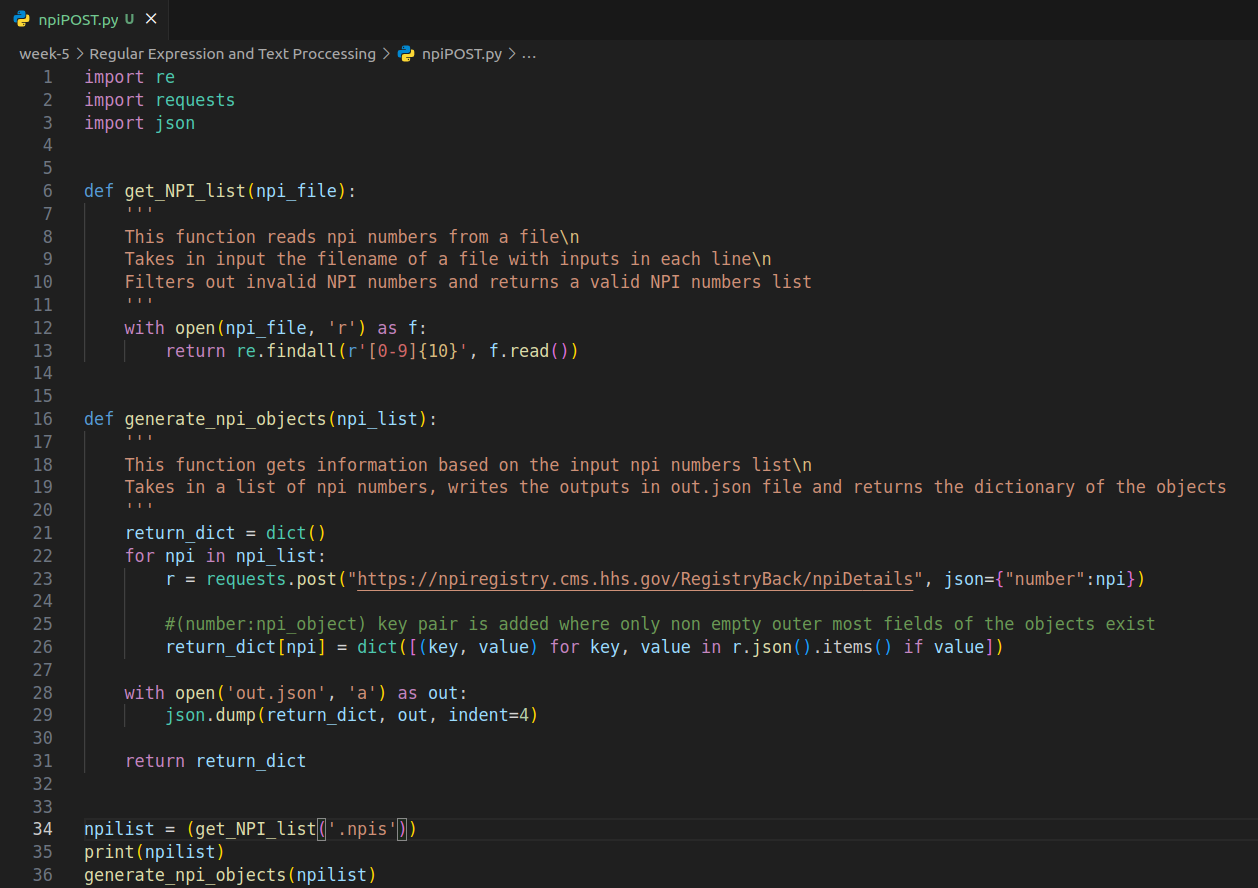
<https://github.com/Deepanshu-TTN/bootcamp-git/tree/master/week-5/Regular%20Expression%20and%20Text%20Proccessing>

Q1)Learn about “National Provider Identifier (NPI)” https://en.wikipedia.org/wiki/National\_Provider\_Identifier

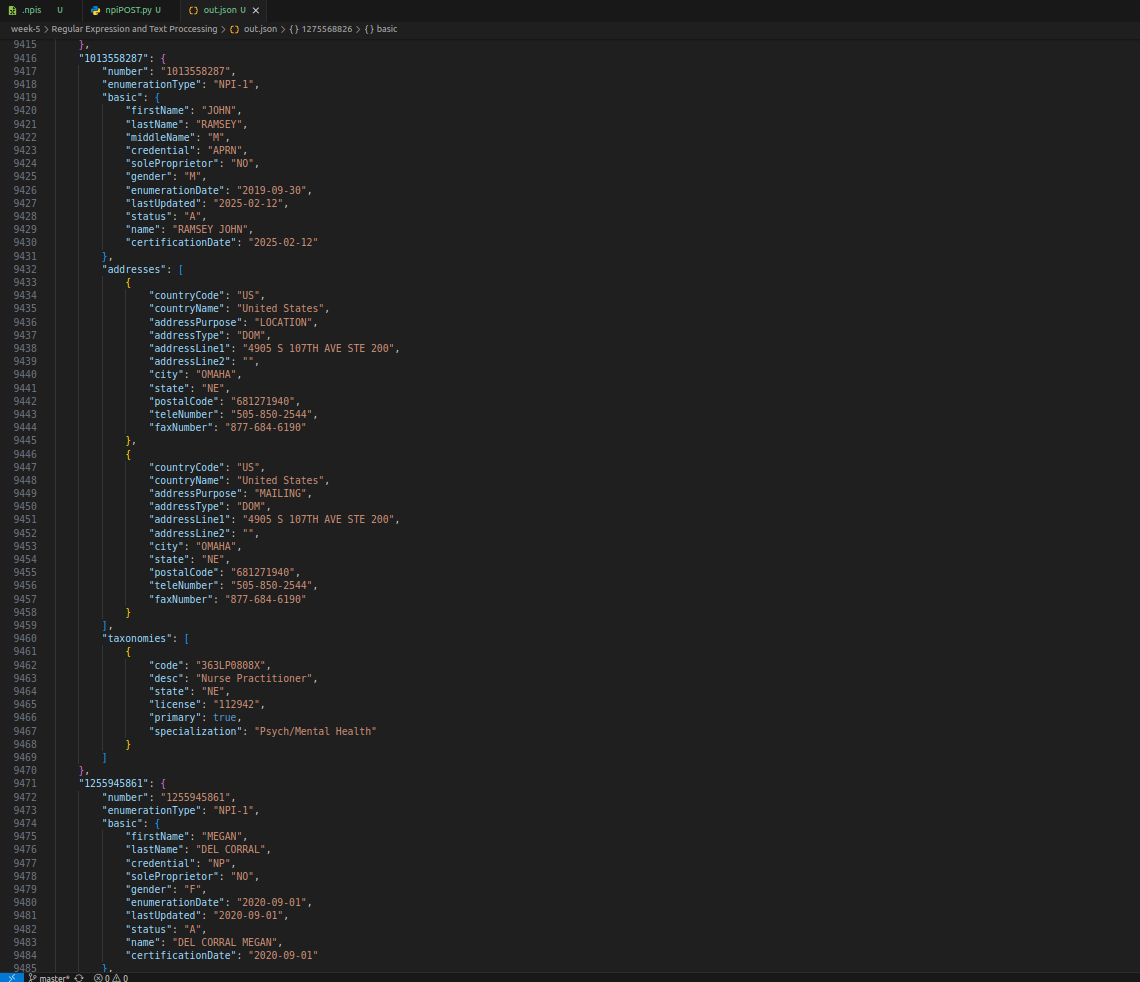
On the portal https://npiregistry.cms.hhs.gov/search, one can search for details associated with an NPI. For example, use 1114473527 for the ‘NPI Number’ input field and click on the “Search” button

The task in this assignment is to create a Python API to fetch details corresponding to an NPI as JSON.

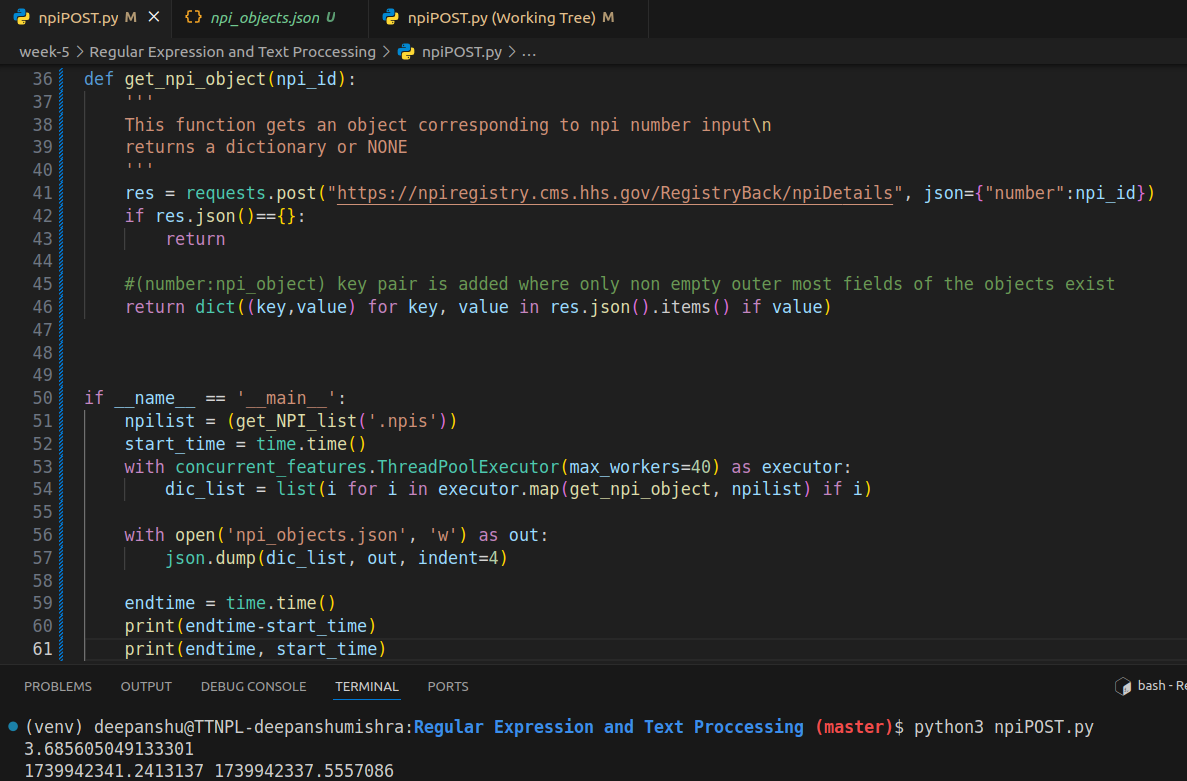
Here the .npis file contains all the inputs given in the bootcamp question



out.json file with all the objects



Update: added a concurrent threads approach for better execution time



Updated the previous function to dump a list as well for consistent output json



Fermoa Plant is an online portal from where one can order different types of sansevierias, either in combo or as individual:

https://fermosaplants.com/collections/sansevieria

 This assignment is about scrapping the portal to prepare an Excel (not CSV) dataset.  Go through all pages of items: “https://fermosaplants.com/collections/sansevieria?page=<number>”   Visit each item and download the following information in Excel.  \* 1 Item in 1 row \* Columns:  <URL> <TYPE> <Price> <Number> <Verigated> <name1> <name2> <name3> ……<nameN>

URL: link of page

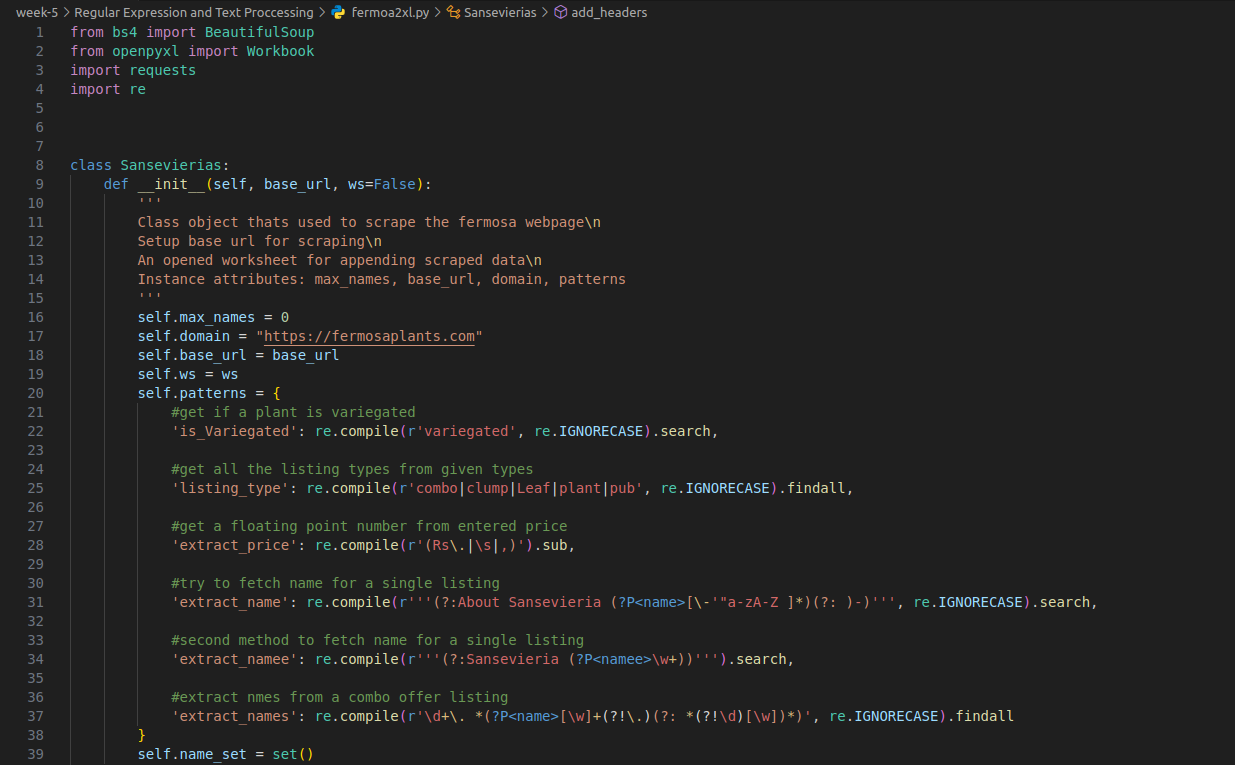
Type: combo, clump, Leaf, plant, pub

Price: Price in rupees Number: number of plant in combo

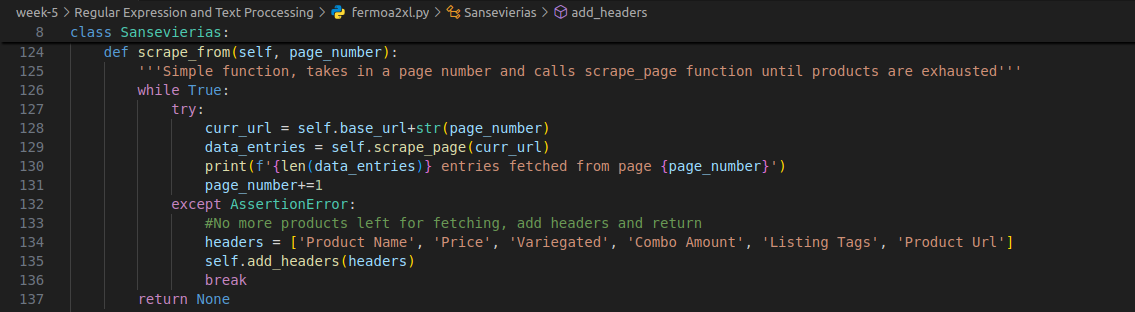
Name: Plant names (one plant name in one column)

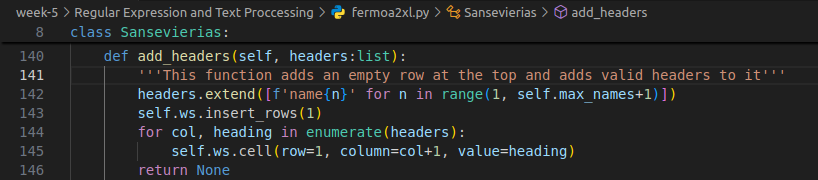
The Sansevierias class

This class is initialized by a base listing url and a worksheet object, upon creation a few regex patterns are compiled too for later use, also added a set to store unique names fetched using this object



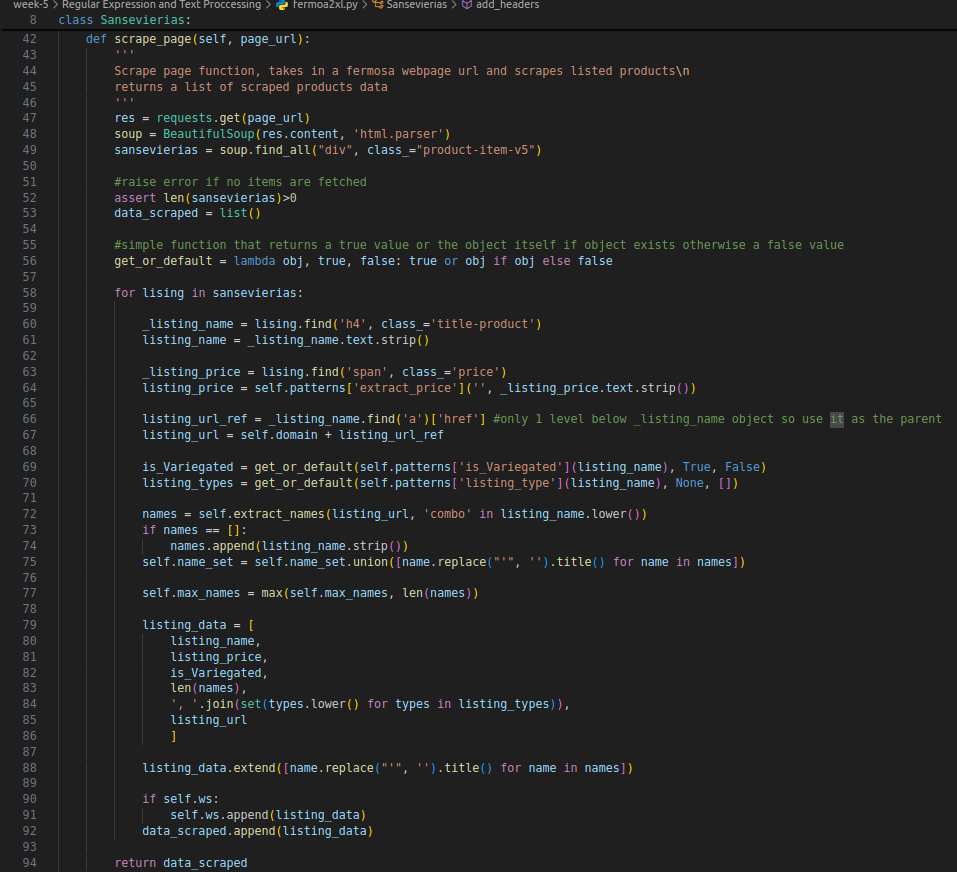
The scrape\_from method continuously calles the scrape\_page function from an initial page number value till the website is exhausted demonstrated by the AssertionError which then calls a add header method after done scraping

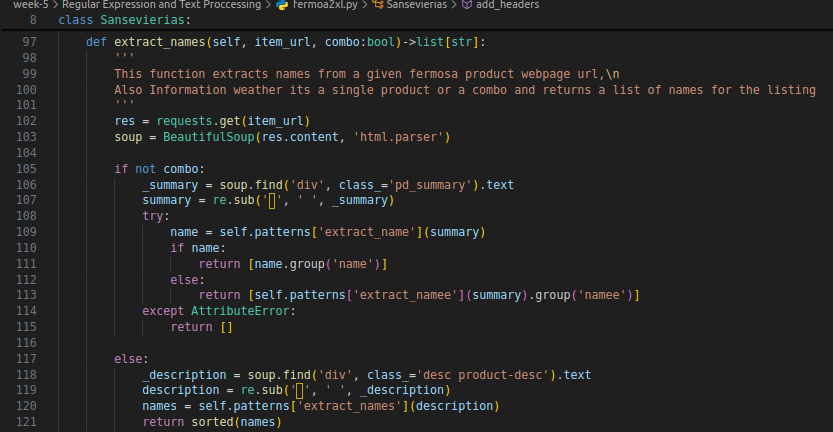




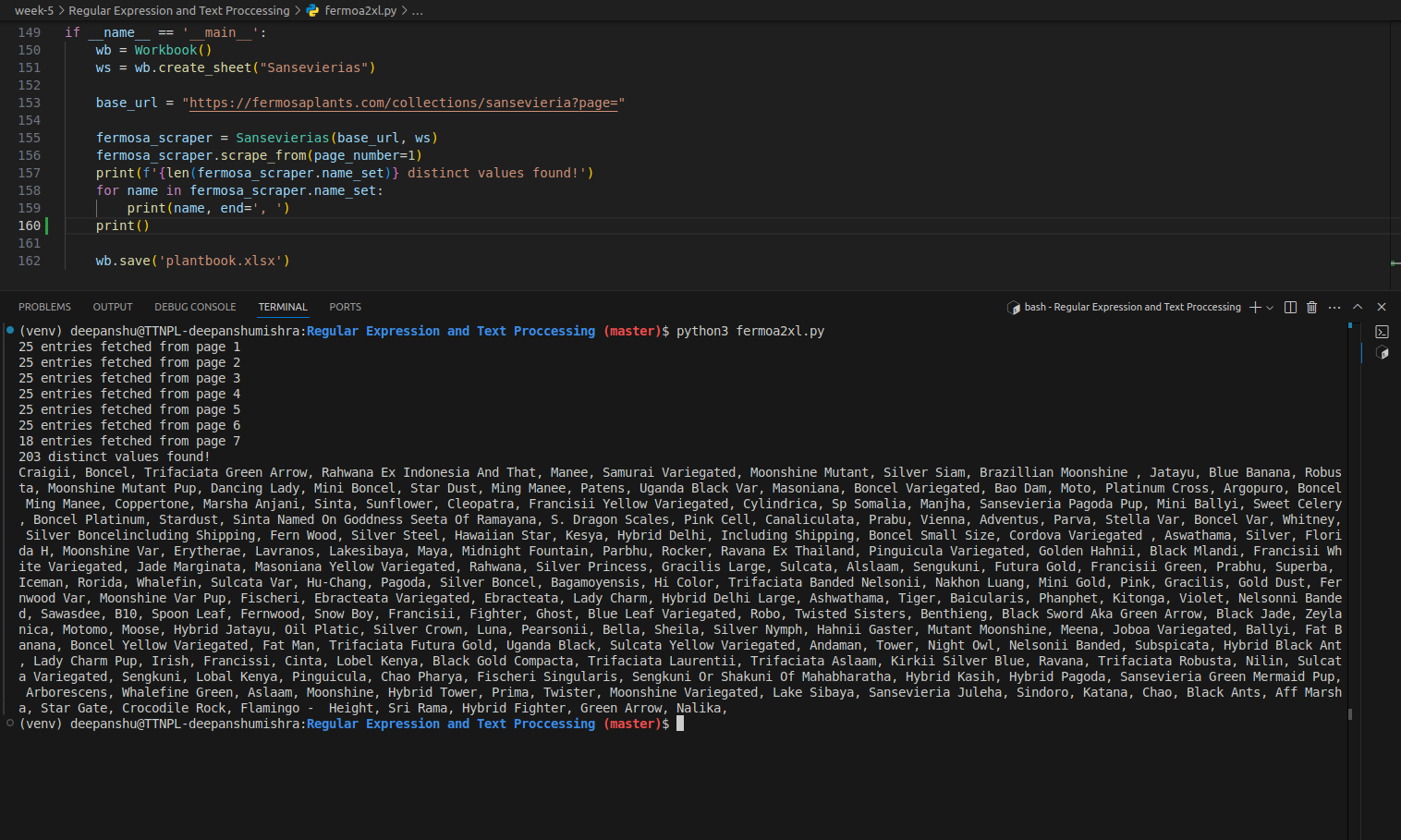
The scrape\_page() method takes in a page\_url and tries to add product details to the worksheet object

* assert len(sansevierias)>0 : raises assertion error which has to be handled in an outer function makes sure we are not stuck scraping empty pages
* get\_or\_default() method takes an object, if the object is nonetype it returns the ‘false’ value passed otherwise returns ‘true’ value or the object itself
* Listing\_data is the list holding information about individual listings derived from the code above it.
* Also names list is fetched using extract\_names() method and after fetching it updates the max\_names (for headers) and name\_set(distinct name values) of the current instance



The extract\_names function returns a list of names from given listing url, if product is combo or not is also passed through the parameters, re.sub is cleaning the string (removing a weird space character)

Example working:



Creates a new workbook, with a new sheet and saves it after scraping

Output Data link: <https://docs.google.com/spreadsheets/d/1qrd-Qp2fW6AAqktQ9-c76rIy9NhqCBzLUnqZkKosN_0/edit?usp=sharing>

snapshot:

