

Programming and Data Structures with Python Lab

Lab5. Dictionaries and Regular Expressions

Question1. Write a program for Fruit Inventory Management.

1. Create a dictionary **fruits** with fruit name as key and quantity available as values. There are 20 apples, 50 bananas, 100 oranges. Then, print outputs for the following queries.
2. Show the entire dictionary fruits (Print output as apples -> 20, bananas -> 50, etc)
3. How many bananas are there?
4. How many items in the dictionary?
5. Does grapes available in the dictionary?
6. Does pears exists in the dictionary?. If so, return its quantity, otherwise, add 10 pears to dictionary.
7. Show all fruit names in ascending order (Iterate using for loop)
8. Show all fruits in descending order of quantities
9. Remove pears from the dictionary.
10. Develop a function **show()** that displays fruit name and quantity (Use .format() for pretty printing)
11. Develop a function **add_fruit(name, quantity)** that receives fruit name and quantity as input and increases the quantity of the fruit. Then, display the current inventory by calling **show()**.
12. Now, add 40 apples to inventory by calling **add_fruit(name, quantity)**
13. Now, add 100 bananas to inventory, by calling **add_fruit(name, quantity)**
14. Now, show the current inventory, by calling **show()**
15. Write the inventory fruits onto a file. (Use **Pickle** for file writing and reading)
16. Now, open Pickle file and display the inventory.

Question2. Write a program for Telephone Directory Management

1. Create an empty dictionary called **customers**, where name is a key and contacts is a list of contacts such as phoneno and email ID for each customer.
2. Ask user to enter name and his contacts for N customers. Add them to dictionary **customers**. Stop reading when user types "done".
3. Show the contacts for customer "rex". If not exists, print message "Contacts not exists.."
4. Add a new customer with name "rex", phone number 9942002764 and email id raj कुमार@bhc.edu
5. Show all customers both name and contacts. (Use items() method, unpack it and print inside **for** loop)
6. Show all customer contacts (Iterate using for loop)
7. Show all customer names in alphabetical order
8. How many customers are there in your dictionary?
9. Remove customer "rex" from dictionary customers

Question3. Write a program for Character and word counter.

- Develop an application that reads a file and prints words in descending order of their frequency.
- Also print the letters such as 'a', 'b', etc, in decreasing order of frequency. Your program should convert all the input to lower case and only count the letters a-z. Your program should not count spaces, digits, punctuation, or anything other than the letters a-z.

Question4. Using Email Collections file, mbox-short.txt, write a python program for the following queries.

1. Search for lines that contain 'From' and print them
2. Search for lines that start with 'From' and print them
3. Search for lines that start with 'F', followed by 2 characters, followed by 'm:'
4. Search for lines that start with From and have an at sign and print them
5. Search for lines that have an at sign between characters and print them (Use findall())
6. Search for lines that have an at sign between characters. The characters must be a letter or number and print them

7. Search for lines that start with 'X' followed by any non white space characters and ':', followed by a space and any number. The number can include a decimal.
8. Search for lines that start with 'X' followed by any non whitespace characters and ':' followed by a space and any number. The number can include a decimal. Then print the number if it is greater than zero.
9. Search for lines that start with 'Details: rev=', followed by numbers and '.'. Then print the number if it is greater than zero
10. Search for lines that start with From and a character followed by a two digit number between 00 and 99 followed by ':'. Then print the number if it is greater than zero

Question5. Baby Names Popularity Analysis

Reference: <https://developers.google.com/edu/python/exercises/baby-names>

The Social Security administration has this neat data by year of what names are most popular for babies born that year in the USA. The files baby1990.html baby1992.html ... contain raw html pages. Take a look at the html and think about how you might scrape the data out of it.

In the babynames.py file, implement the extract_names(filename) function which takes the filename of a baby1990.html file and returns the data from the file as a single list -- the year string at the start of the list followed by the name-rank strings in alphabetical order. ['2006', 'Aaliyah 91', 'Abigail 895', 'Aaron 57', ...].

Modify `main()` so it calls your `extract_names()` function and prints what it returns (main already has the code for the command line argument parsing). If you get stuck working out the regular expressions for the year and each name, solution regular expression patterns are shown at the end of this document. Note that for parsing webpages in general, regular expressions don't do a good job, but these webpages have a simple and consistent format.

Rather than treat the boy and girl names separately, we'll just lump them all together. In some years, a name appears more than once in the html, but we'll just use one number per name. Optional: make the algorithm smart about this case and choose whichever number is smaller.

Build the program as a series of small milestones, getting each step to run/print something before trying the next step. This is the pattern used by experienced programmers -- build a series of incremental milestones, each with some output to check, rather than building the whole program in one huge step.

Printing the data you have at the end of one milestone helps you think about how to re-structure that data for the next milestone. Python is well suited to this style of incremental development. For example, first get it to the point where it extracts and prints the year and calls `sys.exit(0)`. Here are some suggested milestones:

- Extract all the text from the file and print it
- Find and extract the year and print it
- Extract the names and rank numbers and print them
- Get the names data into a dict and print it
- Build the `[year, 'name rank', ...]` list and print it
- Fix `main()` to use the `ExtractNames` list

Earlier we have had functions just print to standard out. It's more re-usable to have the function `*return*` the extracted data, so then the caller has the choice to print it or do something else with it. (You can still print directly from inside your functions for your little experiments during development.)

Have `main()` call `extract_names()` for each command line arg and print a text summary. To make the list into a reasonable looking summary text, here's a clever use of `join`: `text = '\n'.join(mylist) + '\n'`

The summary text should look like this for each file:

```
2006
Aaliyah 91
Aaron 57
Abigail 895
Abbey 695
Abbie 650
...
```


