Programming and Data Structures with Python Lab

Lab5. Dictionaries and Regular Expressions

```
235229143
swetha K
```

```
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 graphs 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.
```

In [1]:
```
with open('pickle.txt','w') as file:
    fruits={"apple":20,"banana":50,"oranges":100}
    f1="nos of {} => {}".format("apple",20)
    f2="nos of {} => {}".format("banana",50)
    f3="nos of {} => {}".format("oranges",100)
    print(f1)
    print(f2)
    print(f3)
    print("the number of bananas:",fruits["banana"])
    print("The number of items in the dictionary:",len(fruits))
    if "grape" in fruits:
        print("Yes, grape is inside the dictionary.")
    else:
        print("NO, grape is not inside the dictionary.")
    if "pear" in fruits:
        print("Yes, pear is inside the dictionary.")
    else:
        fruits.update({"pear":10})
        print("Quantity of pears in dictionary:",fruits["pear"])
    for j in sorted(fruits):
        print(j)
    for z in reversed(fruits.values()):
        print(z)
    fruits.pop("pear")
    def show():
        print(fruits)
    def add_fruit(a,b):
        fruits[a]=b
        print(fruits)
    i=0
    while i<2:
        a=input("Enter the fruit name:")
        b=input("Enter the fruit's quantity:")
        print(add_fruit(a,b))
        i=i+1
    show()
f=open('pickle.txt')
```

```
nos of apple => 20
nos of banana => 50
nos of oranges => 100
the number of bananas: 50
The number of items in the dictionary: 3
NO, grape is not inside the dictionary.
Quantity of pears in dictionary: 10
apple
banana
oranges
pear
10
100
50
20
Enter the fruit name:apple
Enter the fruit's quantity:40
{'apple': '40', 'banana': 50, 'oranges': 100}
None
Enter the fruit name:banana
Enter the fruit's quantity:50
{'apple': '40', 'banana': '50', 'oranges': 100}
None
{'apple': '40', 'banana': '50', 'oranges': 100}
```

```
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
rajkumar@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
```

In [5]:
```
def TDM():
    customers = {}

    while True:
        name = input("Enter customer name (or 'done' to stop): ")
        if name == "done":
            break

        phone_no = input("Enter phone number: ")
        email = input("Enter email ID: ")

        contacts = {
            "phoneno": phone_no,
            "email": email
        }

        customers[name] = contacts

    print("\nContacts for customer 'rex':")
    if "rex" in customers:
        print("Phone:", customers["rex"]["phoneno"])
        print("Email:", customers["rex"]["email"])
    else:
        print("Contacts not exist.")

    customers["rex"] = {
        "phoneno": "9942002764",
        "email": "rajkumar@bhc.edu"
    }

    print("\nAll customers and contacts:")
    for name, contacts in customers.items():
        print("Name:", name)
        print("Phone:", contacts["phoneno"])
        print("Email:", contacts["email"])
        print()

    print("\nAll customer contacts:")
    for name, contacts in customers.items():
        print("Name:", name)
        print("Phone:", contacts["phoneno"])
        print("Email:", contacts["email"])
        print()

    print("\nCustomer names in alphabetical order:")
    for name in sorted(customers.keys()):
        print(name)

    print("\nNumber of customers:", len(customers))

    if "rex" in customers:
        del customers["rex"]
        print("\n'rex' removed from the dictionary.")
TDM()
```

```
Enter customer name (or 'done' to stop): ramesh
Enter phone number: 3848483483
Enter email ID: ram@mail.com
Enter customer name (or 'done' to stop): rex
Enter phone number: 9942002764
Enter email ID: rajkumar@bhc.edu
Enter customer name (or 'done' to stop): done

Contacts for customer 'rex':
Phone: 9942002764
Email: rajkumar@bhc.edu

All customers and contacts:
Name: ramesh
Phone: 3848483483
Email: ram@mail.com

Name: rex
Phone: 9942002764
Email: rajkumar@bhc.edu


All customer contacts:
Name: ramesh
Phone: 3848483483
Email: ram@mail.com

Name: rex
Phone: 9942002764
Email: rajkumar@bhc.edu


Customer names in alphabetical order:
ramesh
rex

Number of customers: 2

'rex' removed from the dictionary.
```

```
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.
```

In [4]:
```
import re
from collections import Counter

def collectorder():
    filename = input("Enter the filename: ")

    try:
        with open(filename, 'r') as file:
            text = file.read()
            text = text.lower()

            # Remove non-alphabet characters and split into words
            words = re.findall(r'\b[a-z]+\b', text)

            word_counter = Counter(words)
            letter_counter = Counter(text)

            print("\nWords in descending order of frequency:")
            for word, count in word_counter.most_common():
                print(f"{word}: {count}")

            print("\nLetters in descending order of frequency:")
            for letter, count in letter_counter.most_common():
                if letter.isalpha():
                    print(f"{letter}: {count}")

    except FileNotFoundError:
        print("File not found.")
    except Exception as e:
        print("An error occurred:", str(e))
collectorder()
```

```
Enter the filename: prasath.txt

Words in descending order of frequency:
hi: 1
this: 1
is: 1
prasath: 1
from: 1
trichy: 1

Letters in descending order of frequency:
h: 4
i: 4
t: 3
s: 3
r: 3
a: 2
p: 1
f: 1
o: 1
m: 1
c: 1
y: 1
```

```
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
```

In [2]:
```
import re

def analyse():
    filename = 'mbox-short.txt'  # Make sure the file is in the same directory

    try:
        with open(filename, 'r') as file:
            lines = file.readlines()

            # Query 1: Search for lines that contain 'From'
            print("Lines that contain 'From':")
            for line in lines:
                if 'From' in line:
                    print(line.rstrip())

            print("\n-------------------------\n")

            # Query 2: Search for lines that start with 'From'
            print("Lines that start with 'From':")
            for line in lines:
                if line.startswith('From'):
                    print(line.rstrip())

            print("\n-------------------------\n")

            # Query 3: Search for lines that start with 'F', followed by 2 chai
            print("Lines that match 'F__m:' pattern:")
            for line in lines:
                if re.search(r'F..m:', line):
                    print(line.rstrip())

            print("\n-------------------------\n")

            # Query 4: Search for lines that start with From and have an at sig
            print("Lines starting with 'From' and containing '@':")
            for line in lines:
                if re.search(r'^From.*@', line):
                    print(line.rstrip())

            print("\n-------------------------\n")

            # Query 5: Search for lines that have an at sign between characters
            print("Lines with at sign between characters:")
            for line in lines:
                matches = re.findall(r'\S+@\S+', line)
                if matches:
                    print(matches)

            print("\n-------------------------\n")

            # Query 6: Search for lines with at sign between letters or numbers
            print("Lines with at sign between letters or numbers:")
            for line in lines:
                matches = re.findall(r'[a-zA-Z0-9]+@[a-zA-Z0-9]+', line)
                if matches:
                    print(matches)
            print("\n-------------------------\n")

            # Query 7: Search for lines that start with 'X' followed by any no
            # followed by a space and any number (decimal included). Print the
            print("Numbers after 'X__:' pattern:")
            for line in lines:
                matches = re.findall(r'X\S+: (\d+(\.\d+)?)', line)
                for match in matches:
                    if float(match[0]) > 0:
                        print(match[0])

            print("\n-------------------------\n")

            # Query 8: Search for lines that start with 'X' followed by any no
            # followed by a space and any number (decimal included). Print the
            print("Numbers after 'X__:' pattern (alternate approach):")
            for line in lines:
                match = re.search(r'X\S+: (\d+(\.\d+)?)', line)
                if match and float(match.group(1)) > 0:
                    print(match.group(1))

            print("\n-------------------------\n")

            # Query 9: Search for lines that start with 'Details: rev=' follow
            # Print the number if it's greater than zero.
            print("Numbers after 'Details: rev=':")
            for line in lines:
                match = re.search(r'Details: rev=([\d.]+)', line)
                if match and float(match.group(1)) > 0:
                    print(match.group(1))

            print("\n-------------------------\n")

            # Query 10: Search for lines that start with From and a character
            # followed by ':'. Print the number if it's greater than zero.
            print("Numbers after 'From __ __:' pattern:")
            for line in lines:
                match = re.search(r'From [a-zA-Z]([0-9]{2}):', line)
                if match and int(match.group(1)) > 0:
                    print(match.group(1))

    except FileNotFoundError:
        print("File not found.")
    except Exception as e:
        print("An error occurred:", str(e))
analyse()
```

```
Lines that contain 'From':
From prasath1234@gmail.com
From namashivayam1234@gmail.com
From xander11@mail.com
From prsath and namachi

------------------------

Lines that start with 'From':
From prasath1234@gmail.com
From namashivayam1234@gmail.com
From xander11@mail.com
From prsath and namachi

------------------------

Lines that match 'F__m:' pattern:

------------------------

Lines starting with 'From' and containing '@':
From prasath1234@gmail.com
From namashivayam1234@gmail.com
From xander11@mail.com

------------------------

Lines with at sign between characters:
['prasath1234@gmail.com']
['namashivayam1234@gmail.com']
['xander11@mail.com']

------------------------

Lines with at sign between letters or numbers:
['prasath1234@gmail']
['namashivayam1234@gmail']
['xander11@mail']

------------------------

Numbers after 'X__:' pattern:

------------------------

Numbers after 'X__:' pattern (alternate approach):

------------------------

Numbers after 'Details: rev=':

------------------------

Numbers after 'From __ __:' pattern:
```

```
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', 'Abagail 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 restructure 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
Abagail 895
Abbey 695
Abbie 650
...
```

In [10]:
```
import re
def extract_names(filename):
    # Read the content of the HTML file
    with open(filename, 'r') as file:
        html_content = file.read()

    # Extract the year from the filename
    year_match = re.search(r'baby(\d{4})\.html', filename)
    if year_match:
        year = year_match.group(1)
    else:
        print(f"Could not find the year in the filename: {filename}")
        return None

    # Extract the name-rank strings using regular expressions
    names_rank = re.findall(r'<td>(\d+)</td><td>(\w+)</td><td>(\w+)</td>', htm

    # Build the final list with year and name-rank strings
    baby_names_data = [year]
    for rank, boy_name, girl_name in names_rank:
        baby_names_data.append(f"{boy_name} {rank}")
        baby_names_data.append(f"{girl_name} {rank}")

    return baby_names_data

def main():
    filenames = "baby1992.html baby1990.html"
    filenames = filenames.split()

    for filename in filenames:
        baby_names_data = extract_names(filename)
        if baby_names_data:
            print('\n'.join(baby_names_data))
main()
```

```
1992
Michael 1
Ashley 1
Christopher 2
Jessica 2
Matthew 3
Amanda 3
Joshua 4
Brittany 4
Andrew 5
Sarah 5
Brandon 6
Samantha 6
Daniel 7
Emily 7
Tyler 8
Stephanie 8
James 9
Elizabeth 9
```