LAB REPORT 1: Python Basics and Conditions

- **In this lab we learn Python basics and conditions.**
- **Python is high level programming language.**
- **Its syntax easy to understand.**
- **Its object oriented programming language.**
- **Let see the python basics& conditions.**
- **In this exercise we use python conditions like a if and if else control statements**
- **And applying some mathematical methods**
- **That satisfy the given question**
- **And find the solution easily**
- **The result is given and get the output from jupyter notebook.**

LAB REPORT 2.PYTHON LOOPS

- **A loop statement allows us to execute a statement or group of statements multiple times.**
- **In this lab we learn for loops and while loops**
- **The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true. We generally use this loop when we don't know the number of times to iterate beforehand.**
- **We use a while loop in this lab**
- **And satisfy the question use of while loop and conditions**
- **No difficulty was faced in this lab**
- **And finally get the output from jupyter notebook.**

LAB REPORT3 : Python functions and Modules

- **In this lab create function and some logical and mathematical programs will do this lab.**
- **Grouping related code into a module makes the code easier to understand and use.**
- **A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.**
- **In this exercise using functions and modules**
- **And satisfy the question and give a solution**
- **We learn function and conditional statement in this lab**
- **And finally get a output from jupyter notebook.**

LAB REPORT 4. Python string processsing

- **In this lab we use a function and character and user defined function and loops**
- **Using this concepts. Its help to satisfy the question and get the output from jupyter notebook**
- **In this exercise we apply a some logics like a digits,vowels,consonants and remove punctuation.**
- **In this lab .we can use the join() method to concatenate two strings.we will find all the methods that a string object can call.**
- **And finally get the output from jupyter noteook.**

LAB REPORT 5. List processing in python

- **In this lab already covered concepts and function. We will use . but in this exercise we will see the tuple,for loop and using invoke methods.**
- **Using other string functions like first(),sort_first(),middle().**
- **In this exercise we learn so much of functions and operations.**
- **Use of this method and function satisfy the questions**
- **And write a program and apply a method and operations.**
- **Finally get the output from jupyter notebook.**

LAB REPORT 6: Python File Processing

- **In this lab we use file processing method and concept will cover.**
- **Creating a Text File. Before we can begin working in Python, we need to make sure we have a file to work.**
- **We learn file processing methods and open,read,close function can used.**
- **And apply needed methods and function.**
- **The file processing was learning in this exercise.**
- **Finally get a output from jupyter notebook.**

LAB REPORT 7. Dictionaries in python

- **In this lab we will see the dictionary concept**
- **Use of dictionary and satisfy the question and given conditions.**
- **Dictionaries are used to store data values in key:value pairs.**
- **Write a program for fruit inventory management and telephone directory management and write a character and word counter.**
- **Some logic are working and get a results.**
- **And get outputs from jupyter notebook.**

LAB REPORT 8: Python regular expressions

- **For this exercise we used regular expression and apply some logics**
- **"re" module provides regular expression support. The re.search() method takes a regular expression pattern and a string and searches for that pattern within the string.**
- **Using email collections file and program will satisy the following queries and conditions.**
- **And open a mbox file and satisfy the conditions.**
- **Extract the file and get a solution**
- **And get a outputs. help of jupyter notebook.**

LAB REPORT 10.Implementation of Map, filter and Reduce Functions.

- **In this lab we will do some functions.**
- **And create a user defined functions and implementation of Map,filter and Reduce functions.**

- **Apply a function and conditions are needed places**
- **We learn some ideas from learned concepts.**
- **Get some reference from internet especially stack overflow**
- **Stack overflow is used for so many places in our python coding.**
- **And finally a get output from jupyter notebook.**

LAB REPORT 11: Retrieving Data From Web and Parsing.

- **In this lab some libraries are import help of anaconda prompt.**
- **In this exercise retrieve data from web page using URLLIB.**
- **And also import some libraries like Bs4 and beautifulsoup.**
- **Its may help supporting the packages and webpages.**
- **And additionally used html tags for web scraping**
- **Web scraping is the process of collecting and parsing raw data from the Web, and the Python community has come up with some pretty powerful web scraping tools.**
- **In some websites are there how to webscraping using python**
- **Also our lab manual given a some references and websites.**
- **We learn webscraping and usage of libraries**
- **And finally write the code and get a results and outputs from jupyter notebook.**

LAB REPORT 12: Data base Programming using sqlite3

- **In this exercise we will see important concept.Its very useful for web development.**
- **How to connect data base use of sqlite3 or mysql database**
- **And also learn CRUD operation like a create,read,update and delete.**

- **Python is easy for webdevelopment and storing a data help of sqldatabase.**
- **Import some library like a sqlite3. its used to access to allow the connection. We learn data base access.**
- **And after create,insert,delete,update functions are used.**
- **In this lab was most usefull for python web development.**
- **And finally the database connected and stored the values in sqlite3**

LAB REPORT 13. 2D and 3D Data visualization using Matplotlib And Seaborn

- **For this lab we learn data visualization. And how to using Matplotlib and seaborn**
- **Seaborn is another of Python's data visualization library built on top of Matplotlib, which have a high-level interface with attractive designs.**
- **Plot library is used for creating interactive and multidimensional plots making the process of data analysis easier by providing a better visualization for the data.**
- **In this lab we will face some difficulties.**
- **But finally we can do and get results.**

LAB REPORT 14 : Animated Data visualization using R

- **For this lab we learn Animated Data visualization**
- **And use a R-language and gganimate package to plot various animated charts using sample datasets.**
- **We will see how to visualize animated bar chart,line chart and scatter plot using R and gganimate package.**
- **The packages that are required to build animated plots in R are: ggplot2.**

- **For first it will be difficult but we learn the structure and usage.**
- **Finally Get a animated data visualization and get a barchart,line chart and scatter plot using R and gganimate package.**

Lab1. Python Basics and Conditions

Question1. Write a program in Python to input length and breadth of a rectangle and print the area and perimeter of it.

* Test you code with atleast 2 test cases

In [5]:

```
l=int(input("Length : "))
b=int(input("Breadth : "))
area=l*b
perimeter=2*(l+b)
print("Area of Rectangle : ",area)
print("Perimeter of Rectangle : ",perimeter)
```

```
Length : 12
Breadth : 12
Area of Rectangle :  144
Perimeter of Rectangle :  48
```

Question2. Write a program, which accepts annual basic salary of an employee and calculates and displays the Income tax as per the following rules.

If Basic is less than Rs. 1,50,000/-, then Tax = 0.
If Basic is from Rs.1,50,000/- to Rs. 3,00,000/-, then tax is 20%.
If Basic is greater than Rs.3,00,000/-, then tax is 30%.
Print name, annual income and tax.
Write test cases to validate all conditions

In [8]:

```
e_name=input("Employee Name: ")
salary=float(input("Your Annual Basic Salary: "))
if(salary<150000):
    tax=0
    netsalary=salary-tax
    print("No taxable Amount")
    print("Your Annual Income: ",netsalary)
elif(salary<300000):
    tax=0.2*salary
    netsalary=salary-tax
    print("Your Tax Amount: ",tax)
    print("Your Annual Income: ", netsalary)
elif(salary>300000):
    tax=0.3*salary
    netsalary=salary-tax
    print("Your Tax Amount: ",tax)
    print("Your Annual Income: ", netsalary)
```

```
Employee Name: Deebug
Your Annual Basic Salary: 10000000
Your Tax Amount:  3000000.0
Your Annual Income:  7000000.0
```

Question3. Write a program to accept quantity and rate for

three (3) items. Compute the total sales amount. Also compute and print the discount as follows:

Amount > Rs. 2000/- : 20% discount
Amount between Rs. 1500/- to Rs.1999/- :15% discount
Amount between Rs. 1000/- to Rs.1499/- 8 % discount
Compute final amount to be paid.
Print name, rate and quantity of 3 items. Then print total sales amount, total dis
count and final amount to be paid to shop.
Write 3 test cases to validate all conditions

In [6]:

```
print("PSPRDEPARTMENT STORE")
name_item1=str(input("enter the item1: "))
quantity_1=int(input("enter the quantity1: "))
rate_1=int(input("enter the rate1: "))

name_item2=str(input("enter the item2: "))
quantity_2=int(input("enter the quantity2: "))
rate_2=int(int(input("enter the rate2: ")))

name_item3=str(input("enter the item3: "))
quantity_3=int(input("enter the quantity3: "))
rate_3=int(input("enter the rate3: "))

amount=(quantity_1*rate_1)+(quantity_2*rate_2)+(quantity_3*rate_3)
price=amount

if amount > 2000:
    print("DISCOUNT 20%.....||\n")
    discount=(amount*20)/100
    price -= discount
elif amount > 1500 and amount < 1999:
    print("DISCOUNT 15%.....||\n")
    discount=(amount*15)/100
    price -= discount
elif amount > 1000 and amount < 1499:
    print("DISCOUNT 8%.....||\n")
    discount=(amount*8)/100
    price -= discount
print("\ntotal amount: ",amount)
print("\ntotal discount: ",discount)
print("\nfinal price to be paid by customer: ",price)
```

```
PSPRDEPARTMENT STORE
enter the item1: rice
enter the quantity1: 12
enter the rate1: 12
enter the item2: sugar
enter the quantity2: 12
enter the rate2: 139
enter the item3: munch
enter the quantity3: 3
enter the rate3: 123
DISCOUNT 20%.....||


total amount:  2181

total discount:  436.2

final price to be paid by customer:  1744.8
```

Question4. Evaluate the expressions using Pen and Paper first and then print the value.

X1=(11+31+23+8+7+5)/((1-(1/2)-(1/20)))
X2=(((10*8)+8-((7//5)%(5**4)))&3)|(2<<1)

In [15]:

```
x1=(11+31+23+8+7+5)/((1-(1/2)-(1/20)))
print(x1)
x2=(((10*8)+8-((7//5)%(5**4)))&3)|(2<<1)
print(x2)
```

```
188.88888888888889
7
```

Question5. Write a program to accept name, marks for three subjects and find the total marks secured, average and also display the class obtained.

Class I - above 80%
Class II - 60% to 80%
Pass class - 40% to 59% and
Fail otherwise #### Print a message as "Congratulations << your name>>, you secured a total of <total>

In [19]:

```
name = str(input("Enter your name: "))
subject_1 = float(input("Please enter Subject 1 Mark: "))
subject_2 = float(input("Please enter Subject 2 Mark: "))
subject_3 = float(input("Please enter Subject 3 Mark: "))

total = int(subject_1+subject_2+subject_3)
average = total / 3
percentage = (total / 300) * 100

if(percentage>=80):
    print("\nCongratulation!", name, "you secured a total of", total, "and your class is
: 1st Class")
elif(percentage>=60 and percentage<=80):
    print("\nCongratulation!", name, "you secured a total of", total, "and your class is
: 2nd Class")
elif(percentage>=40 and percentage<=59):
    print("\nCongratulation!", name, "you secured a total of", total, "and your class is
: Pass Class")
else:
    print("you are fail")
```

```
Enter your name: Deepak
Please enter Subject 1 Mark: 100
Please enter Subject 2 Mark: 89
Please enter Subject 3 Mark: 79

Congratulation! Deepak you secured a total of 268 and your class is: 1st Class
```

Question6. Read a number from keyboard. Print whether it is odd number, even number, positive number, negative number or zero. Also, print if its ASCII value represents a lower case or upper case letter or digit.

Write 8 test cases to validate odd, even, positive, negative, zero, lower case, upper case and digit input types

In [22]:

```
n=int(input("Enter the number to check ODD or EVEN:  \n"))
```

```
if n>0:
    print("number is POSITIVE \n")
    if n%2==0:
        print("number is even  \n")
    else:
        print("number is odd   \n")
elif n<0:
    print("number is NEGETIVE   \n")
    if n%2==0:
        print("number is even   \n")
    else:
        print("number is odd     \n")
else:
    print("number is ZERO   \n")


print("ASCII values   \n")
if n>=65 and n<=91:
    print("represents UPPERCASE LETTER   \n")
elif n>=97 and n<=122:
    print("represents LOWERCASE LETTER   \n")
elif n>=48 and n<=57:
    print("represents DIGIT   \n")

val=chr(n)
print("the value" ,val)
char=str(input("Enter ASCII values \n"))
print(ord(char))
n=100
chr(n)

char=str(input("Enter a single number or a letter \n"))
if(ord(char)>=65 and ord(char)<=90):
    print("IT IS AN UPPERCASE LETTERS")
elif(ord(char)>=97 and ord(char)<=122):
    print("IT IS AN LOWERCASE LETTERS")
else:
    (ord(char)>=48 and ord(char)<=57)
    print("IT IS A DIGIT")
```

```
Enter the number to check ODD or EVEN:
99
number is POSITIVE

number is odd

ASCII values

represents LOWERCASE LETTER

the value c
Enter ASCII values
k
107
Enter a single number or a letter
0
IT IS A DIGIT
```

In [ ]:

LAB-2

Question1. Write a program that accepts numbers continuously as long as the number is positive and prints the sum of the numbers read (Use while loop). A sample user interaction will be:

```
Enter a number: 2
Enter a number: 1
Enter a number: 4
Enter a number: 6
Enter a number: -10
Sum = 13
```

In [2]:

```
total = 0
i = int(input("Enter any integer "))
while i > 0:
    total = total + i
    i = int(input("Enter next integer "))
print("\nThe sum of all numbers entered is", total)
```

```
Enter any integer 1
Enter next integer 2
Enter next integer 3
Enter next integer 4
Enter next integer 5
Enter next integer -2

The sum of all numbers entered is 15
```

Question 2. Write a program to take the values of two integers m and n from the user. Calculate the sum of even number between m and n (including both m and n). Please note that value of m must be less than value of n. If m > n. then you must print a message “Value of m should be less than n” and ask for next input values. Print the values of m, n and sum. (Use while loop). The program should continue until user types ‘q’ to quit the program.

Sample user interaction:

```
Enter m: 1
Enter n: 10
Sum of even numbers: 20
Do you want to quit (Type q)? :
Enter m: 2
Enter n: 10
Sum of even numbers: 20
Do you want to quit (Type q)? :
Enter m: 20
Enter n: 10
Value of m should be less than n
Do you want to quit (Type q)? : q
```

In [3]:

```
while True:
```

```
    m = int(input("Enter any value for m= "))
    n = int(input("Enter any value for n= "))

    print("Value of m should be less then n")

    sum = 0
    for i in range (m,n+1):
        if i%2 ==0:
            sum = sum + i
    print(sum)
    reply=str(input("\n Do you want to quit? (type 'q'): "))
    if reply =='q':
        break;
```

```
Enter any value for m= 1
Enter any value for n= 10
Value of m should be less then n
30

 Do you want to quit? (type 'q'): y
Enter any value for m= 2
Enter any value for n= 10
Value of m should be less then n
30

 Do you want to quit? (type 'q'): q
```

Question3. Write a program to accept n and display its multiplication table. Value of n must be provided by the user. (Example: n 1, n 2,....,n*10) (Use for loop)

In [14]:

```
num = int(input("Enter the number: "))
print("Multiplication Table of", num)
for i in range(1, 11):
    print(num,"X",i,"=",num * i)
```

```
Enter the number: 123
Multiplication Table of 123
123 X 1 = 123
123 X 2 = 246
123 X 3 = 369
123 X 4 = 492
123 X 5 = 615
123 X 6 = 738
123 X 7 = 861
123 X 8 = 984
123 X 9 = 1107
123 X 10 = 1230
```

Question4. Write a program that receives an integer and prints the sum of its digits. For example, an input 125 will print output 1+2+5=8.

Try out with the following test cases

125

12

2

-15

In [16]:

```
num = int(input("Enter a integer: "))
total = 0
while num > 0:
```

```
        digit = num % 10
        total = total + digit
        num = num // 10
print("Sum of the integer is:", total)
```

```
Enter a integer: 125
Sum of the integer is: 8
```

Question5. Develop an application in Python that repeatedly reads numbers until the user enters done. Once done is entered, print out the total, count, and average of the numbers. If the user enters anything other than a number, detect their mistake using try and except and print an error message and skip to the next number.

In [18]:

```
num = 0
tot = 0
while True:
    number = input("Enter a number: ")
    if number == 'done':
        break
    try :
        num1 = int(number)
    except:
        print('Invailed Input')
        continue
    num = num+1
    tot = tot + num1
    avg = tot/num
print ('\ndone')
print ('\ntotal',tot)
print('\nnum',num)
print('\naverage',avg)
```

```
Enter a number: 1
Enter a number: 2
Enter a number: 3
Enter a number: 4
Enter a number: 3
Enter a number: 12
Enter a number: done

done

total 25

num 6

average 4.166666666666667
```

In [ ]:

```
```

Lab3. Python Functions and Modules

Question 1: create a function prime() that receives an integer and returns whether n is prime or not. Print all prime numbers from 1 to 100 by calling prime() function.

In [34]:

```
def prime(num):
    for i in range(2,num):
        if num%i == 0:
            return 0
    else:
        return num

while True:
    n=int(input("Enter the number: "))
    if n==0:
        print("\nLoop ends here... \n\n\n\nHere we are printing all prime numbers from 1
to 100 using prime function: \n\n")
        break
    a=prime(n)
    if a!= 0:
        print(n,"is prime number\n")
    else:
        print(n,"is not prime number\n")

def prime(n):
    status = True
    if n < 2:
        status = False
    else:
        for i in range(2,n):
            if n % i == 0:
                status = False
    return status

for n in range(1,100):
    if prime(n):
        if n==100:
            print ('Prime',n)
        else:
            print ('Prime',n)
```

```
Enter the number: 1
1 is prime number

Enter the number: 2
2 is prime number

Enter the number: 3
3 is prime number

Enter the number: 4
4 is not prime number

Enter the number: 5
5 is prime number

Enter the number: 6
6 is not prime number

Enter the number: 0

Loop ends here...
```

```
Here we are printing all prime numbers from 1 to 100 using prime function:


Prime 2
Prime 3
Prime 5
Prime 7
Prime 11
Prime 13
Prime 17
Prime 19
Prime 23
Prime 29
Prime 31
Prime 37
Prime 41
Prime 43
Prime 47
Prime 53
Prime 59
Prime 61
Prime 67
Prime 71
Prime 73
Prime 79
Prime 83
Prime 89
Prime 97
```

Explaination 1: Here we need to print the integer from the user wheather the entered number is prime number or not to do that we are creating a function using (def) and giving certian conditions in (while, if, else) which to validate specific condition in each block. Then we call prime function to print prime numbers from 1 to 100...

Question 2: Develop a simple arithmetic calculator for 4 operations. The program should continue calculation until user types 'q' to quit. A sample user interaction can be:

Enter operator(q to quit): + Enter value 1: 10 Enter value 2: 20 Result = 30

Create 4 functions add(), subtract(), multiply(), and divide() that receives two values and returns the result of the operation. Now, perform the following operations by calling the corresponding functions. Validate your outputs.

1. **10+20**
2. **20-5**
3. **8*5**
4. **50/3**

In [35]:

```
def add(a,b):
    return a+b
def sub(a,b):
    return a-b
def mul(a,b):
    return a*b
def div(a,b):
    if b==0:
        print("division is not possible ")
    else:
        return a/b

while True:
    print("Enter the opertor: ")
    operator=input()
    if (operator=='q'):
        break
    a=int(input("\nEnter the value of 1: "))
    b=int(input("Enter the value of 2: "))
```

```
    if (operator=='+'):
        r=add(a,b)
    elif (operator=='-'):
        r=sub(a,b)
    elif (operator =='*'):
        r=mul(a,b)
    elif (operator=='/'):
        r=div(a,b)
    print("Result is ",r,'\n')
print("Quit")
```

```
Enter the opertor:
+

Enter the value of 1: 10
Enter the value of 2: 20
Result is  30

Enter the opertor:
-

Enter the value of 1: 20
Enter the value of 2: 5
Result is  15

Enter the opertor:
*

Enter the value of 1: 8
Enter the value of 2: 5
Result is  40

Enter the opertor:
/

Enter the value of 1: 50
Enter the value of 2: 3
Result is  16.666666666666668

Enter the opertor:
q
Quit
```

Explaination 2: Here we are creating 4 seperate functions using 1.add 2.sub 3.mul 4.div I need to get input from the user to 'enter the operator', 'Enter the value 1:', 'Enter the value 2:' then using if and else operation it perform a validate add, sub, mul, div while performing division if the input is 0 then it print division is not possible else it perform next step. Then finally when the user enter q the program is to quite.

Question3.Create a function factorial() that takes an integer and returns its factorial value.

- **You can create as a non-recursive version of factorial.**
- **Also, check factorial of negative number does not exist.**
- **Factorial of 0 is 1.**
- **Save this Python file as factorial_definition.py.**

Now, open another file and can import factorial_definition.py as follows:

- **import factorial_definition**
- **You can call function function as factorial_definition.factorial().**

Now, print the following factorial values:

- **1. factorial_definition.factorial(3)**
- **1. factorial_definition.factorial(5)**
- **1. factorial_definition.factorial(10)**

In [1]:

```
def factorial(n):
    result = 1;
    if(n < 0):
        print("The factorial does not exist for negative numbers")
    elif(n==0):
        print("The factorial of 0 is 1")
    elif(n==1):
        return 1
    else:
        for i in range(2, n + 1):
            result = result * i;
        return result;
```

Create a non recursive function factorial() that takes an integer and returns its factorial value. Save the python file as .py and import the module and use the functions.

In [2]:

```
import factorial_definition

factorial_definition.factorial(3)
```

Out[2]:

```
6
```

In [3]:

```
factorial_definition.factorial(5)
```

Out[3]:

```
120
```

In [4]:

```
factorial_definition.factorial(10)
```

Out[4]:

```
3628800
```

Lab4. Python String Processing

Question1:Develop a function count_letter(string, search) that returns the number of times search character appears in a string.

Test cases:

```
Str = "hello world".Search = 'o'. Calling count_letter(str, search) should return
output 2
Str = "HeLlo wOrld". Search = 'o'. Then, calling count_letter(str, search) will ret
urn output 1
```

Modify count_letter() so that it ignores case sensitivity, so that o and O are same.

```
Str = "HeLlo wOrld". Search='o'. Calling count_lettern(str, search) will return out
put 2
```

In [2]:

```
def count_letter(word, search):
    return count
word = input("Enter the words to search: ").lower()
search = input("Enter the character to search: ").lower()
count = 0
for char in word:
    if char == search:
        count += 1
print(count_letter(word, search))
```

```
Enter the words to search: hello world
Enter the character to search: o
2
```

Question2.Write a program that counts the number of spaces, digits, vowels and consonants in a string that the user inputs. Print the string, no of spaces, no of digits, no of vowels and no of consonants.

Test case: Enter a string: Bishop Heber College 17. Then output should be:

```
Given string: Bishop Heber College 17
No. of spaces: 3
No. of digits: 2
No. of vowels: 7
No. of consonants: 12
```

In [4]:

```
s =input("Enter the string: ")
a = s.lower()
vowels = "aeiou"
consonants = "bcdfghjklmnpqrstvwxyz"
digits = "1234567890"
space = " "
c = 0
v = 0
d = 0
sp = 0
```

```
for i in a:
    if i in vowels:
        v+=1
    elif i in consonants:
        c+=1
    elif i in digits:
        d+=1
    elif i in space:
        sp+=1

print("\nSpace: ",sp,"\nDigits: ",d,"\nVowels: ",v,"\nConsonants: ",c)
```

```
Enter the string: bishop heber college 19

Space:  3
Digits:  2
Vowels:  7
Consonants:  11
```

Question3.Develop a function remove_punctuation(str) that returns the string afer removing the following punctuations.

Punctuation List = "!\"#$%&'()*+,-./:;<=>?@[]^{|}"

Test case:

Str="Bishop's College !....". Calling remove_punctuation(str) should return output as "Bishop College"
Str="#bhc trending @cs $placements::>." Calling remove_punctuation(str) should return output as "bhc trending cs placements"

In [6]:

```
def remove_punctuation(string):
    punctuations = '''!()-[];:'"\,<>./?@#$%^&*_'''
    for x in string.lower():
        if x in punctuations:
            string = string.replace(x, "")
    print(string)
```

In [7]:

```
string=(input("Enter the string with symbols: "))
remove_punctuation(string)
```

```
Enter the string with symbols: #bhc trending @cs $placements::>.
bhc trending cs placements
```

Question4: Write a program that asks the user for a word. Translate their word into Pig Latin. Pig Latin game takes the first consonant (or set of first consonants) of an English word, moves it to the end of the word and suffixes an ay. If the first letter is a vowel, do not move that vowel,but instead add "way" at the end of the word.

Test Cases:

Enter a word: pig
    Output: ig-pay
Enter a word: banana
    Output: anana-bay
Enter a word: trash
    Output: ash-tray
Enter a word: apple
    Output: apple-way

```
Enter a word: orange
    Output: orange-way
```

Modify your program so that it becomes a function piglatin(word) and returns translated word as output. Call this function 3 times with the same inputs and validate the outputs.

In [9]:

```
word = input("Enter a word to translate to pig latin:")
def piglatin(word):
    ay = 'ay'
    way = 'way'
    p = '-'
    consonant = ('B','C','D','F','G','H','J','K','L','M','N','P','Q','R','S','T','Y','V',
'X','Z')
    vowel = ('A','E','I','O','U')
    first_letter = word[0]
    first_letter = str(first_letter)
    first_letter = first_letter.upper()
    if first_letter in consonant:
        print(first_letter,'is a consonant')
        length_of_word = len(word)
        remove_first_letter = word[1:length_of_word]
        pig_latin = remove_first_letter+p+first_letter+ay
        print('The word in Pig Latin is:',pig_latin)
    elif first_letter in vowel:
        print(first_letter,"is a vowel")
        pig_latin = word+p+way
        print('The word in Pig Latin is:',pig_latin)
    else:
        print('I dont know what',first_letter,'is')
```

```
Enter a word to translate to pig latin:pig
```

In [10]:

```
piglatin(word)
```

```
P is a consonant
The word in Pig Latin is: ig-Pay
```

In [ ]:

```

```

Lab5. List Processing in Python

Question1:Write a function find_average(student) that takes student tuples as input and print student rollno, name, marks and average marks as output.

- **Test Cases:**

```
1.Stud1=(1,"rex",60,85,70)
find_average(stud1)
```

Modify the above function find_average(student) so that it processes a tuple of tuples.

```
2.Stud2=(2,"rex",(80,75,90))
find_average(stud2)
```

In [3]:

```
def find_average(student):
    roll,name,marks = student
    total = 0
    for mark in marks:
        total += mark
    avg = total / len(marks)
    print("Roll No:",roll,"Name:",name,"Average:",avg)
```

In [4]:

```
stud1 = (1,"rex",(60,85,70))
```

In [5]:

```
find_average(stud1)
```

```
Roll No: 1 Name: rex Average: 71.66666666666667
```

In [6]:

```
stud2 = (2,"Rex",(80,75,90))
```

In [7]:

```
find_average(stud2)
```

```
Roll No: 2 Name: Rex Average: 81.66666666666667
```

Question2:Write a weight management program that prompts that prompts the user to enter in 7 days of their body weight values as float numbers. Store them in list. Then print first day weight, last day weight, 4th day weight, highest weight, lowest weight and average weight.

Finally, print if average weight < lowest weight, then print "Your weight management is excellent". Otherwise print "Your weight management is not good. Please take care of your diet".

In [5]:

```
list=[]
for i in range(7):
    c=float(input("Enter your weight:"))
    list.append(c)
print("\n\nFirst day weight",list[0])
print("\nLast day weight",list[-1])
print("\nHighest weight is",max(list))
print("\nLowest weight is",min(list))
print("\nAverage of weight is",round(sum(list)/len(list)))
```

```
Firstday=list[0]
Lastday=list[-1]
avg=round(sum(list)/len(list))
low_weight=min(list)
if(avg<low_weight):
    print("\nYour weight management is excellent")
else:
    print("\nYour weight management is not good. Please take care of your diet")
```

```
Enter your weight:70
Enter your weight:71
Enter your weight:72
Enter your weight:73
Enter your weight:74
Enter your weight:75
Enter your weight:76


First day weight 70.0

Last day weight 76.0

Highest weight is 76.0

Lowest weight is 70.0

Average of weight is 73

Your weight management is not good. Please take care of your diet
```

QUESTION 3: Write a function lastN(lst,n) that takes a list of integers and n and returns n largest numbers.

How many numbers you want to enter?: 6 Enter a number: 12 Enter a number: 32 Enter a number: Enter a number: Enter a number: Enter a number:

How many largest numbers you want to find?: 3 Largest numbers are: 52, 45, 32

In [13]:

```
def lastN(lst, n):
    final_list =[]

    for i in range(n):
        max1 = 0

        for j in lst:
            if j> max1:
                max1 = j;

        lst.remove(max1);
        final_list.append(max1)
    print("\nlargest numbers:",final_list)

b=int(input("How many number want to enter?: "))
lst=[]
for i in range(b):
    c=int(input("\nEnter a number: "))
    lst.append(c)

n=int(input("\nHow many largest numbers you want to find?: "))
lastN(lst,n)
```

```
How many number want to enter?: 6

Enter a number: 12

Enter a number: 34

Enter a number: 18
```

```
Enter a number: 24

Enter a number: 2

Enter a number: 29

How many largest numbers you want to find?: 3

largest numbers: [34, 29, 24]
```

QUESTION 4: Given a list of strings, return a list with the strings in sorted order, except group all the strings that begin with "X" first. Hint: this can be done by making 2 lists and sorting each of them before combining them.

Test Cases:

- **1.Input: ['min','xyz','apple','xanadu','aardvark'] Output:['xanadu','xyz','aardvark','apple','mix']**
- **2.Input: ['ccc','bbb','aaa','xcc','xaa'] Output:['xaa','xcc','aaa','bbb','ccc']**
- **3.Input: ['bbb','ccc','axx','xzz','xaa'] Output: ['xaa','xzz','axx','bbb','ccc']**

In [14]:

```
def front_x(words):
    xlist = []
    alist = []

    for word in words:
        if word.startswith("x"):
            xlist.append(word)
        else:
            alist.append(word)
    return sorted(xlist) + sorted(alist)
```

In [15]:

```
words=['min','xyz','apple','xanadu','aardvark']
front_x(words)
```

Out[15]:

```
['xanadu', 'xyz', 'aardvark', 'apple', 'min']
```

In [16]:

```
words=['ccc','bbb','aaa','xcc','xaa']
front_x(words)
```

Out[16]:

```
['xaa', 'xcc', 'aaa', 'bbb', 'ccc']
```

In [17]:

```
words=['bbb','ccc','axx','xzz','xaa']
front_x(words)
```

Out[17]:

```
['xaa', 'xzz', 'axx', 'bbb', 'ccc']
```

QUESTION 5: Develop a function sort_last(). Given a list of non-empty tuples, return a list sorted in increasing order by the last element in each tuple. Hint: use a custom key= function to extract the last element from each tuple.

Test Cases:

- **1. Input:[(1,7),(1,3),(3,4,5),(2,2)] Output:[(2,2),(1,3),(3,4,5),(1,7)]**
- **1. Input:[(1,3),(3,2),(2,1)] Output:[(2,1),(3,2),(1,3)]**
- **1. Input:[(2,3),(1,2),(3,1)] Output:[(3,1),(1,2),(2,3)]**

In [18]:

```
def last(n): return n[-1]

def sort_list_last(tuples):
  return sorted(tuples, key=last)

print("Test Case:1 \n\tInput:[(1,7),(1,3),(3,4,5),(2,2)] \n\tOutput:",sort_list_last([(1,
7),(1,3),(3,4,5),(2,2)]))

print("Test Case:2 \n\tInput:[(1,3),(3,2),(2,1)] \n\tOutput:",sort_list_last([(1,3),(3,2
),(2,1)]))

print("Test Case:3 \n\tInput:[(2,3),(1,2),(3,1)] \n\tOutput:",sort_list_last([(2,3),(1,2
),(3,1)]))
```

```
Test Case:1
 Input:[(1,7),(1,3),(3,4,5),(2,2)]
 Output: [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
Test Case:2
 Input:[(1,3),(3,2),(2,1)]
 Output: [(2, 1), (3, 2), (1, 3)]
Test Case:3
 Input:[(2,3),(1,2),(3,1)]
 Output: [(3, 1), (1, 2), (2, 3)]
```

Question6. Other String Functions

- **a) Define a function first() that receives a tuple and returns its first element**
- **b) Define a function sort_first() that receives a list of tuples and returns the sorted**
- **c) Print lists in sorted order**
- **d) Define a function middle() that receives a a tuple and returns its middle element**
- **e) Define a functino sort_middle() that receives a list of tuples and returns it sorted using the key middle**
- **f) Print the list [(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)] in sorted order. Output should be: [(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]**

In [22]:

```
def first(num):
    for j in num:
        a,b,c=j
        print(a)

def sort_first(num):
    new_list=sorted(num)
    return new_list

def lists(num):
    return sorted(num, key=None, reverse=0)


def middle(num):
    for j in num:
        a,b,c=j
        print(b)

def sort_middle(num):
    return sorted(num,key = lambda mid:mid[1])
num=[(1,2,3),(2,1,4),(10,7,15),(20,4,50),(30,6,40)]
```

In [23]:

```
first(num)
```

```
1
2
10
20
```

```
30
```

In [25]:

```
sort_first(num)
```

Out[25]:

```
[(1, 2, 3), (2, 1, 4), (10, 7, 15), (20, 4, 50), (30, 6, 40)]
```

In [21]:

```
sort_middle(num)
```

Out[21]:

```
[(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]
```

In [24]:

```
middle(num)
```

```
2
1
7
4
6
```

QUESTION7: Develop a function remove_adjacent(). Given a list of numbers, return a list where all adjacent same elements have been reduced to a single element. You may create a new list or modify the passed in list.

Test Cases:

- **1.Input:[1,2,2,3] and Output:[1,2,3]**
- **2.Input:[2,2,3,3,3,] and Output:[2,3]**
- **3.Input:[].Output:[].**
- **4.Input:[2,5,5,6,6,7] and Output:[2,5,6,7]**
- **5.Input:[6,7,7,8,9,9] and Output:[6,7,8,9]**

In [1]:

```
def remove_adjacent(lst):
    a = []
    for item in lst:
        if len(a):
            if a[-1] != item:
                a.append(item)
        else:
            a.append(item)
    return a
```

In [2]:

```
remove_adjacent([1,2,2,3])
```

Out[2]:

```
[1, 2, 3]
```

In [3]:

```
remove_adjacent([2,2,3,3,3])
```

Out[3]:

```
[2, 3]
```

In [4]:

```
remove_adjacent([2,5,5,6,6,7])
```

Out[4]:

```
[2, 5, 6, 7]
```

In [5]:

```
remove_adjacent([6,7,7,8,9,9])
```

Out[5]:

```
[6, 7, 8, 9]
```

In [6]:

```
remove_adjacent([])
```

Out[6]:

```
[]
```

QUESTION8: Write a function verbing(). Given a string, if its length is at least 3, add'ing' to its end. Unless it already ends in 'ing', in which case add 'ly' instead. If the string length is less than 3, leave it unchanged. Return the resulting string. So 'hail' yields: hailing, 'swimming' yields: swimmingly; 'do' yields:do.

In [7]:

```
def verbing(str):
    length = len(str)
    if length > 2:
        if str[-3:] == 'ing':
            str += 'ly'
        else:
            str += 'ing'
    return str
```

In [8]:

```
verbing("hail")
```

Out[8]:

```
'hailing'
```

In [9]:

```
verbing("swimming")
```

Out[9]:

```
'swimmingly'
```

In [10]:

```
verbing("do")
```

Out[10]:

```
'do'
```

QUESTION9: Develop a function not_bad(). Given a string, find the first appearance of the substring 'not' and 'bad'. If the 'bad' follows the 'not', replace the whole 'not'..'bad' substring with 'good'. Return the resulting string. So 'This dinner is not that bad! 'Yields: This dinner is good!

In [11]:

```
def not_bad(s):
    snot = s.find('not')
    sbad = s.find('bad')
    if sbad>snot:
        s=s.replace(s[snot:(sbad+3)], 'good')
    return s
```

In [12]:

```
not_bad("This dinner is not really that bad!")
```

Out[12]:

```
'This dinner is good!'
```

Lab6. Python File Processing

Question1:Write a program for Password Management System

- **File creation: Ask user to enter N user names and their passwords. Store usernames andpasswords into a file named “loginfile.txt”. Store each user and password in one line.**
- **File Processing: Write a program that opens your “loginfile.txt” file and reads usernames and passwords from it. Store user names in one list and passwords in another lists.**
- **Querying: ask user to enter user name and password for verification. If they match the values stored in the lists, print a message “Login Successful”. Otherwise print a message “Login Failed, try again”**

In [4]:

```
def register():
    username = input("Please input the first 2 letters of your first name and your birth
year ")
    password = input("Please input your desired password ")
    file = open("loginfile.txt","a")
    file.write(username)
    file.write(" ")
    file.write(password)
    file.write("\n")
    file.close()
    if login():
        print("You are now logged in...")
    else:
        print("You aren't logged in!")

def login():
    username = input("Please enter your username: ")
    password = input("Please enter your password: ")
    for line in open("loginfile.txt","r").readlines(): # Read the lines
        login_info = line.split() # Split on the space, and store the results in a list
of two strings
        if username == login_info[0] and password == login_info[1]:
            print("Correct credentials!")
            return True
    print("Incorrect credentials.")
    return False
```

In [5]:

```
register()
```

```
Please input the first 2 letters of your first name and your birth year deepak
Please input your desired password 987654321
Please enter your username: deepak
Please enter your password: 987654321
Correct credentials!
You are now logged in...
```

In [3]:

```
login()
```

```
Please enter your username: deepak
Please enter your password: 987654321
Correct credentials!
```

Out[3]:

```
True
```

Question2: Write a program for Student Performance Analysis

- **Create a text file, 'marks.txt', with N marks as floating point numbers. Open the file, read marks from it and compute and print the highest mark.**
- **If the user runs the program more than once you should not overwrite the previous text file – simply append the marks to the end of the file.**
- **Modify the above program so that it also prints Top-3 highest marks (Note: you may need to use list concept)**
- **Modify the above program so that it also prints the Lowest-3 marks.**

In [4]:

```
marks= [99.0,100.0,95.0,96.0,97.0]
with open('marks1.txt', 'a') as file:
    for mark in marks:
        file.write("%.1f\n" % mark)
number_list=[]
with open('marks1.txt', 'r') as fp:
    number_list = [float(item) for item in fp.readlines()]
print(max(number_list))

def Nmaxelements(list1, N):
    final_list = []
    for i in range(0, N):
        max1 = 0
        for j in range(len(list1)):
            if list1[j] > max1:
                max1 = list1[j];
        list1.remove(max1);
        final_list.append(max1)
    print(final_list)
Nmaxelements(number_list,3)

def Nminelements(list1, N):
    final_list =[];
    for i in range(0, N):
        min1 = 9999999;
        for j in range(len(list1)):
            if list1[j]<min1:
                min1 = list1[j];
        list1.remove(min1);
        final_list.append(min1)
    print(final_list)
Nminelements(number_list,3)
```

```
100.0
[100.0, 100.0, 100.0]
[95.0, 95.0, 95.0]
```

Question:3 Write a program for Stock Price Analysis

- **File Creation: Continually prompt a user for stock name, followed by price values for 5 days. Each row indicates stock name and daily prices of one stock. Store these values in a text file called “stock-prices.txt”. Open the file in Append Mode. Prompt message “Do you want to continue? “ and stop reading values accordingly. Then, you can close your file.**
- **File Processing: Now, open your file for processing. Print stock name, minimum price, maximum price and average price values.**
- **You can also print which day stock price was lowest in the week and which day stock price was highest. So, modify your print statement to print stock name, minimum price & day of minimum price, maximum price & day of maximum price and average price values. (Hint: Use enumerate to get index values)**

In [11]:

```
while True:
    st_name=str(input("Enter the name:  "))
    file=open("stock_prices.txt","a")
    file.write(st_name)
    file.write("  ")
```

```
    for i in range(5):
        p=input()
        file.write(p)
        file.write("  ")
    file.write("\n")
    con = str(input("want to continue  :  "))
    if con =='n':
        break
file.close()
```

```
Enter the name:  HDFC
12
15
20
19
22
want to continue  :  Y
Enter the name:  TESLA
99
100
121
132
143
want to continue  :  Y
Enter the name:  GOOGLE
112
122
143
154
164
want to continue  :  N
Enter the name:  WALMART
11
15
16
25
30
want to continue  :  n
```

In [13]:

```
for st in open("stock_prices.txt","r").readlines():
    p_min=[]
    calc=st.split()
    print(calc[0])
    for i in range(1,4):
        p_min.append(int(calc[i]))
    print(min(p_min))
    print(max(p_min))
    av=sum(p_min)
    avg=av/5
    print(avg)
    print("\n")
```

```
dell
100
121
66.4


apple
132
137
80.6


lenovo
111
122
70.8
```

```
tcl
89
99
55.6


HDFC
12
20
9.4


TESLA
99
121
64.0


GOOGLE
112
143
75.4


WALMART
11
16
8.4
```

In [15]:

```
for st in open("stock_prices.txt","r").readlines():
    p_min=[]
    print("-------------------------")
    calc=st.split()
    print(calc[0])
    for i in range(1,4):
        p_min.append(int(calc[i]))
    mip=min(p_min)
    mxp=max(p_min)
    im=p_min.index(mip)
    ix=p_min.index(mxp)
    print("min price ",mip," on day ",im+1)
    print("max price ",mxp," on day ",ix+1)
```

```
-------------------------
dell
min price  100  on day  1
max price  121  on day  2
-------------------------
apple
min price  132  on day  2
max price  137  on day  3
-------------------------
lenovo
min price  111  on day  1
max price  122  on day  3
-------------------------
tcl
min price  89  on day  1
max price  99  on day  2
-------------------------
HDFC
min price  12  on day  1
max price  20  on day  3
-------------------------
TESLA
min price  99  on day  1
max price  121  on day  3
-------------------------
```

```
GOOGLE
min price  112  on day  1
max price  143  on day  3
-------------------------
WALMART
min price  11  on day  1
max price  16  on day  3
```

Question:4 Write a program for File Explorer

Display the contents of file 1.Count the number of lines in a text file. (Use splitlines()) 2.Count the number of unique words in a file. 3.Find frequency of words in a given file. (Hint: Use Counter object) 4.Show a random line in a file. (Use Random object)

In [25]:

```
print("1.Display the contents of File:")
print("-------")
f = open("samplemv.txt",'r')
display = f.read()
print(display)
f.close()
print("")

print("2.Count the number of lines in a text file:")
print("-------")
file = open("samplemv.txt","r")
Counter = 0
Content = file.read()
CoList = Content.split("\n")

for i in CoList:
    if i:
        Counter += 1
print("Number of lines in the text file:",Counter)
print("\n")


print("3.Count the number of unique words in a file:")
print("-------")
num_words = 0
c = open("samplemv.txt", 'r')
for line in c:
    words = line.split()
    num_words += len(words)
print("Number of words:",num_words)
c.close()
print("\n")


print("4.Find Find frequency of words in a given file:")
print("--------")
fname = input('Enter the file name: ')
print("------------------------------------------------------------------------------
--------------------")
try:
    fhand = open(fname)
    counts = dict()
    for line in fhand:
        words = line.split()
        for word in words:
            if word in counts:
                counts[word] += 1
            else:
                counts[word] = 1
    print(counts)
except:
    print('File cannot be opened:', fname)
print("\n")
```

```
print("5.Show a random line in a file:")
print("--------")
import random
def random_line(fname):
    lines = open(fname).read().splitlines()
    return random.choice(lines)
print(random_line("samplemv.txt"))
```

```
1.Display the contents of File:
-------
Tesla, Inc. is an American electric vehicle and clean energy company based in Palo Alto,
California. Tesla's current products include \
electric cars, battery energy storage from home to grid scale, solar panels and solar roo
f tiles, as well as other related products and
services. Tesla ranked as the world's best-selling plug-in and battery electric passenger
car manufacturer in 2019, with a market share
of 17% of the plug-in segment and 23% of the battery electric segment. Through its subsid
iary SolarCity, Tesla develops and is a major
installer of solar photovoltaic systems in the United States. Tesla is also one of the la
rgest global suppliers of battery energy storage
systems, from home-scale to grid-scale. Tesla installed some of the largest battery stora
ge plants in the world and supplied 1.65 GWh of
battery storage in 2019.
Founded in July 2003 as Tesla Motors, the company's name is a tribute to inventor and ele
ctrical engineer Nikola Tesla. Elon Musk, who
contributed most of the funding in the early days, has served as CEO since 2008. Accordin
g to Musk, the purpose of Tesla is to help
expedite the move to sustainable transport and energy, obtained through electric vehicles
and solar power. Tesla began production of
their first car model, the Roadster, in 2009. The Roadster was a luxury sports car. This
was followed by the Model S sedan in 2012, the
Model X SUV in 2015, and the higher volume Model 3 sedan in 2017. Tesla global vehicle sa
les were 499,550 units in 2020, a 35.8% increase
over the previous year. In 2020, the company surpassed the 1 million mark of electric car
s produced. The Model 3 ranks as the world's
all-time best-selling plug-in electric car, with more than 500,000 delivered.
Tesla has been the subject of numerous lawsuits and controversies arising from statements
and acts of CEO Elon Musk, allegations of
whistleblower retaliation, alleged worker rights violations, and allegedly unresolved and
dangerous technical problems with their
products.

2.Count the number of lines in a text file:
-------
Number of lines in the text file: 17


3.Count the number of unique words in a file:
-------
Number of words: 325


4.Find Find frequency of words in a given file:
--------
Enter the file name: samplemv.txt
--------------------------------------------------------------------------------------
-------------
{'Tesla,': 1, 'Inc.': 1, 'is': 5, 'an': 1, 'American': 1, 'electric': 7, 'vehicle': 2, 'a
nd': 15, 'clean': 1, 'energy': 3, 'company': 2, 'based': 1, 'in': 12, 'Palo': 1, 'Alto,':
1, 'California.': 1, "Tesla's": 1, 'current': 1, 'products': 2, 'include': 1, '\\': 1, 'c
ars,': 1, 'battery': 6, 'storage': 4, 'from': 3, 'home': 1, 'to': 6, 'grid': 1, 'scale,':
1, 'solar': 4, 'panels': 1, 'roof': 1, 'tiles,': 1, 'as': 6, 'well': 1, 'other': 1, 'rela
ted': 1, 'services.': 1, 'Tesla': 9, 'ranked': 1, 'the': 21, "world's": 2, 'best-selling'
: 2, 'plug-in': 3, 'passenger': 1, 'car': 2, 'manufacturer': 1, '2019,': 1, 'with': 3, 'a
': 5, 'market': 1, 'share': 1, 'of': 15, '17%': 1, 'segment': 1, '23%': 1, 'segment.': 1,
'Through': 1, 'its': 1, 'subsidiary': 1, 'SolarCity,': 1, 'develops': 1, 'major': 1, 'ins
taller': 1, 'photovoltaic': 1, 'systems': 1, 'United': 1, 'States.': 1, 'also': 1, 'one':
1, 'largest': 2, 'global': 2, 'suppliers': 1, 'systems,': 1, 'home-scale': 1, 'grid-scale
.': 1, 'installed': 1, 'some': 1, 'plants': 1, 'world': 1, 'supplied': 1, '1.65': 1, 'GWh'
: 1, '2019.': 1, 'Founded': 1, 'July': 1, '2003': 1, 'Motors,': 1, "company's": 1, 'name':
1, 'tribute': 1, 'inventor': 1, 'electrical': 1, 'engineer': 1, 'Nikola': 1, 'Tesla.': 1,
'Elon': 2, 'Musk,': 3, 'who': 1, 'contributed': 1, 'most': 1, 'funding': 1, 'early': 1, '
```

```
days,': 1, 'has': 2, 'served': 1, 'CEO': 2, 'since': 1, '2008.': 1, 'According': 1, 'purp
ose': 1, 'help': 1, 'expedite': 1, 'move': 1, 'sustainable': 1, 'transport': 1, 'energy,'
: 1, 'obtained': 1, 'through': 1, 'vehicles': 1, 'power.': 1, 'began': 1, 'production': 1
, 'their': 2, 'first': 1, 'model,': 1, 'Roadster,': 1, '2009.': 1, 'The': 2, 'Roadster':
1, 'was': 2, 'luxury': 1, 'sports': 1, 'car.': 1, 'This': 1, 'followed': 1, 'by': 1, 'Mod
el': 4, 'S': 1, 'sedan': 2, '2012,': 1, 'X': 1, 'SUV': 1, '2015,': 1, 'higher': 1, 'volume
': 1, '3': 2, '2017.': 1, 'sales': 1, 'were': 1, '499,550': 1, 'units': 1, '2020,': 2, '35
.8%': 1, 'increase': 1, 'over': 1, 'previous': 1, 'year.': 1, 'In': 1, 'surpassed': 1, '1
': 1, 'million': 1, 'mark': 1, 'cars': 1, 'produced.': 1, 'ranks': 1, 'all-time': 1, 'car
,': 1, 'more': 1, 'than': 1, '500,000': 1, 'delivered.': 1, 'been': 1, 'subject': 1, 'nume
rous': 1, 'lawsuits': 1, 'controversies': 1, 'arising': 1, 'statements': 1, 'acts': 1, 'a
llegations': 1, 'whistleblower': 1, 'retaliation,': 1, 'alleged': 1, 'worker': 1, 'rights
': 1, 'violations,': 1, 'allegedly': 1, 'unresolved': 1, 'dangerous': 1, 'technical': 1,
'problems': 1, 'products.': 1}
```

5.Show a random line in a file:

```
electric cars, battery energy storage from home to grid scale, solar panels and solar roo
f tiles, as well as other related products and
```

Question5: Develop an application in Python to read through the email data ("mbox-short.txt") and when you find line that starts with "From", you will split the line into words using the split function. We are interested in who sent the message, which is the second word on the From line: From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008. You will parse the From line and print out the second word for each From line, then you will also count the number of From (not From:) lines and print out a count at the end.

In [118]:

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if line.startswith('From '):
        print(line)
```

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
From louis@media.berkeley.edu Fri Jan  4 18:10:48 2008
From zqian@umich.edu Fri Jan  4 16:10:39 2008
From rjlowe@iupui.edu Fri Jan  4 15:46:24 2008
From zqian@umich.edu Fri Jan  4 15:03:18 2008
From rjlowe@iupui.edu Fri Jan  4 14:50:18 2008
From cwen@iupui.edu Fri Jan  4 11:37:30 2008
From cwen@iupui.edu Fri Jan  4 11:35:08 2008
From gsilver@umich.edu Fri Jan  4 11:12:37 2008
From gsilver@umich.edu Fri Jan  4 11:11:52 2008
From zqian@umich.edu Fri Jan  4 11:11:03 2008
From gsilver@umich.edu Fri Jan  4 11:10:22 2008
From wagnermr@iupui.edu Fri Jan  4 10:38:42 2008
From zqian@umich.edu Fri Jan  4 10:17:43 2008
From antranig@caret.cam.ac.uk Fri Jan  4 10:04:14 2008
From gopal.ramasammycook@gmail.com Fri Jan  4 09:05:31 2008
From david.horwitz@uct.ac.za Fri Jan  4 07:02:32 2008
From david.horwitz@uct.ac.za Fri Jan  4 06:08:27 2008
From david.horwitz@uct.ac.za Fri Jan  4 04:49:08 2008
From david.horwitz@uct.ac.za Fri Jan  4 04:33:44 2008
From stephen.marquard@uct.ac.za Fri Jan  4 04:07:34 2008
From louis@media.berkeley.edu Thu Jan  3 19:51:21 2008
From louis@media.berkeley.edu Thu Jan  3 17:18:23 2008
From ray@media.berkeley.edu Thu Jan  3 17:07:00 2008
From cwen@iupui.edu Thu Jan  3 16:34:40 2008
From cwen@iupui.edu Thu Jan  3 16:29:07 2008
From cwen@iupui.edu Thu Jan  3 16:23:48 2008
```

In [119]:

```
fhand = open("mbox-short.txt")
count = 0
for line in fhand:
    line = line.rstrip()
    if line == "": continue
    words = line.split()
```

```
    if words[0] !="From": continue
    print(words[1])
    count = count+1
print ("There were", count, "lines in the file with From as the first word")
```

```
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
zqian@umich.edu
rjlowe@iupui.edu
zqian@umich.edu
rjlowe@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
gsilver@umich.edu
gsilver@umich.edu
zqian@umich.edu
gsilver@umich.edu
wagnermr@iupui.edu
zqian@umich.edu
antranig@caret.cam.ac.uk
gopal.ramasammycook@gmail.com
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
louis@media.berkeley.edu
ray@media.berkeley.edu
cwen@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
There were 27 lines in the file with From as the first word
```

Question6. Write a program to read and write CSV files

1).File Creation: Create MS Excel file (“student_marks.csv”) with 5 rows of student name, mark1, mark2, mark3, mark4. Use comma to separate each value in a row.

2).File Display: Now, open your CSV file and display the file contents row by row (More information at: https://docs.python.org/3/library/csv.html).

3).File Writing: Now, open (“student_marks.csv”) for writing. Ask user to enter name followed by 4 marks for one new student and write them onto the file.

In [15]:

```
from csv import writer
def append_list_as_row(file_name, list_of_elem):

    with open('student_marks.csv', 'a+', newline='') as write_obj:

        csv_writer = writer(write_obj)

        csv_writer.writerow(list_of_elem)

row_contents = ['Suresh',68,78,89,87,90]
row_contents1 = ['ganesh',68,78,89,87,90]
row_contents2 = ['Harish',68,78,89,87,90]
row_contents3 = ['Rajesh',68,78,89,87,90]
append_list_as_row('student_marks.csv', row_contents)
append_list_as_row('student_marks.csv', row_contents1)
append_list_as_row('student_marks.csv', row_contents2)
append_list_as_row('student_marks.csv', row_contents3)
```

In [16]:

```
import csv
with open('student_marks.csv', newline ='') as csvfile:
```

```
    reader = csv.reader(csvfile, delimiter =' ', quotechar ='|')
    for row in reader:
        print(', '.join(row))
```

```
student, name,mark1,mark2,mark3,mark4,mark5
Johnson,78,56,72,95,77
Tom,89,69,74,90,88
Josephine,90,89,93,78,70
Jerry,89,78,70,88,90
David,90,98,87,89,86
Sam,68,78,89,87,90
Ram,68,78,89,87,90
Ramkumar,68,78,89,87,90
Suresh,68,78,89,87,90
ganesh,68,78,89,87,90
Harish,68,78,89,87,90
Rajesh,68,78,89,87,90
```

Lab7. Dictionaries in Python

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.

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.

In [1]:

```
fruits={"apples":20,"bananas":50,"oranges":100}
```

2. Show the entire dictionary fruits

In [2]:

```
print(fruits)
```

```
{'apples': 20, 'bananas': 50, 'oranges': 100}
```

3. How many bananas are there?

In [3]:

```
print(fruits.get('bananas'))
```

```
50
```

4. How many items in the dictionary?

In [4]:

```
len(fruits)
```

Out[4]:

```
3
```

5. Does graphs available in the dictionary?

In [5]:

```
if "grapes" in fruits:
    print("yes")
else:
    print("No")
```

```
No
```

6. Does pears exists in the dictionary?. If so, return its quantity, otherwise, add 10 pears to dictionary.

In [6]:

```
if "pears" in fruits.keys():
    print(fruits.get('pears'))
else:
    fruits["pears"]=10
print(fruits)
```

```
{'apples': 20, 'bananas': 50, 'oranges': 100, 'pears': 10}
```

7. Show all fruit names in ascending order (Iterate using for loop)

In [7]:

```
for keys in fruits:
    print(sorted(fruits.keys()))
```

```
['apples', 'bananas', 'oranges', 'pears']
['apples', 'bananas', 'oranges', 'pears']
['apples', 'bananas', 'oranges', 'pears']
['apples', 'bananas', 'oranges', 'pears']
```

8. Show all fruits in descending order of quantities

In [8]:

```
print(sorted(fruits.keys(),reverse=True))
```

```
['pears', 'oranges', 'bananas', 'apples']
```

9. Remove pears from the dictionary.

In [9]:

```
del fruits['pears']
print(fruits)
```

```
{'apples': 20, 'bananas': 50, 'oranges': 100}
```

10. Develop a function show() that displays fruit name and quantity (Use .format() for pretty printing)

In [10]:

```
def show():
    print("{} {} {}".format(*fruits.items()))
show()
```

```
('apples', 20) ('bananas', 50) ('oranges', 100)
```

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().

In [11]:

```
def add_fruit(fruits,name,quantity):
    fruits[name]=fruits.get(name,0)+quantity
```

12. Now, add 40 apples to inventory by calling add_fruit(name, quantity)

In [12]:

```
fruits={"apples":20,"bananas":50,"oranges":100}
add_fruit(fruits,'apples',40)
print(fruits)
```

```
{'apples': 60, 'bananas': 50, 'oranges': 100}
```

13. Now, add 100 bananas to inventory, by calling add_fruit(name, quantity)

In [13]:

```
fruits={"apples":20,"bananas":50,"oranges":100}
add_fruit(fruits,"bananas",100)
print(fruits)
```

```
{'apples': 20, 'bananas': 150, 'oranges': 100}
```

14. Now, show the current inventory, by calling show()

In [14]:

```
show()
```

```
('apples', 20) ('bananas', 150) ('oranges', 100)
```

15. Write the inventory fruits onto a file. (Use Pickle for file writing and reading)

In [15]:

```
import pickle
new_inventory={"apple":300,"mango":87,"banana":320}
# Write the dictionary to the pickle file
file = open("fruits_inventory.p", "wb")
pickle.dump(new_inventory, file)
print(new_inventory)
```

```
{'apple': 300, 'mango': 87, 'banana': 320}
```

16. Now, open Pickle file and display the inventory.

In [16]:

```
file = open("fruits_inventory.p","rb")
dict2 = pickle.load(file)
file.close()
print(new_inventory)
```

```
{'apple': 300, 'mango': 87, 'banana': 320}
```

Question 2. 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”

In [17]:

In [17]:

```
customer = {}
while(True):
    row = input("Enter the name and number: ")
    info = row.split()
    if(info[0] == "done"):
        break
    name = info[0]
    phone = info[1]
    email = info[2]
    contacts=[phone,email]
    customer[name] = contacts
print("\nPrinting Contacts:")
for name, phone in customer.items():
    print(name, ":", phone)
```

```
Enter the name and number: abc 7867906567 abc@gmail.com
Enter the name and number: def 9887765904 def@gmail.com
Enter the name and number: ghi 9090907889 ghi@gmail.com
Enter the name and number: done

Printing Contacts:
abc : ['7867906567', 'abc@gmail.com']
def : ['9887765904', 'def@gmail.com']
ghi : ['9090907889', 'ghi@gmail.com']
```

3.Show the contacts for customer “rex”. If not exists, print message “Contacts not exists..”

In [18]:

```
if 'rex' in customer:
    print("rex in present")
else:
    print("There is no such contact")
```

```
There is no such contact
```

4.Add a new customer with name “rex”, phone number 9942002764 and email id yourname@bhc.edu

In [19]:

```
customer['rex']=['9942002764','jonny@bhc.edu']
```

5.Show all customers both name and contacts. (Use items() method, unpack it and print inside for loop)

In [20]:

```
for name, contacts in customer.items():
    print(name,contacts)
```

```
abc ['7867906567', 'abc@gmail.com']
def ['9887765904', 'def@gmail.com']
ghi ['9090907889', 'ghi@gmail.com']
rex ['9942002764', 'jonny@bhc.edu']
```

6.Show all customer contacts (Iterate using for loop)

In [21]:

```
for name in customer:
    print(customer[name])
```

```
['7867906567', 'abc@gmail.com']
['9887765904', 'def@gmail.com']
['9090907889', 'ghi@gmail.com']
['9942002764', 'jonny@bhc.edu']
```

7.Show all customer names in alphabetical order

In [22]:

```
sorted(customer.keys())
```

Out[22]:

```
['abc', 'def', 'ghi', 'rex']
```

8.How many customers are there in your dictionary?

In [23]:

```
len(customer.items())
```

Out[23]:

```
4
```

9.Remove customer “rex” from dictionary customers

In [24]:

```
del customer['rex']
```

In [25]:

```
customer
```

Out[25]:

```
{'abc': ['7867906567', 'abc@gmail.com'],
 'def': ['9887765904', 'def@gmail.com'],
 'ghi': ['9090907889', 'ghi@gmail.com']}
```

Lab8. Python Regular Expressions

Question1: 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

In [16]:

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search ('From:', line):
        print(line)
```

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: gsilver@umich.edu
From: gsilver@umich.edu
From: zqian@umich.edu
From: gsilver@umich.edu
From: wagnermr@iupui.edu
From: zqian@umich.edu
From: antranig@caret.cam.ac.uk
From: gopal.ramasammycook@gmail.com
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: louis@media.berkeley.edu
From: ray@media.berkeley.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
```

2.Search for lines that start with 'From' and print them

In [15]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:', line):
        print(line)
```

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: gsilver@umich.edu
From: gsilver@umich.edu
From: zqian@umich.edu
```

```
From: gsilver@umich.edu
From: wagnermr@iupui.edu
From: zqian@umich.edu
From: antranig@caret.cam.ac.uk
From: gopal.ramasammycook@gmail.com
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: louis@media.berkeley.edu
From: ray@media.berkeley.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
```

3.Search for lines that start with 'F', followed by 2 characters, followed by 'm:'

In [17]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^F..m:', line):
        print(line)
```

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: gsilver@umich.edu
From: gsilver@umich.edu
From: zqian@umich.edu
From: gsilver@umich.edu
From: wagnermr@iupui.edu
From: zqian@umich.edu
From: antranig@caret.cam.ac.uk
From: gopal.ramasammycook@gmail.com
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: louis@media.berkeley.edu
From: ray@media.berkeley.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
```

4.Search for lines that start with From and have an at sign and print them

In [18]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:.+@', line):
```

```
        print(line)
```

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: gsilver@umich.edu
From: gsilver@umich.edu
From: zqian@umich.edu
From: gsilver@umich.edu
From: wagnermr@iupui.edu
From: zqian@umich.edu
From: antranig@caret.cam.ac.uk
From: gopal.ramasammycook@gmail.com
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: david.horwitz@uct.ac.za
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: louis@media.berkeley.edu
From: ray@media.berkeley.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
From: cwen@iupui.edu
```

5.Search for lines that have an at sign between characters and print them (Use findall())

In [19]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('\S+@\S+', line)
    if len(x) > 0:
        print(x)
```

```
['stephen.marquard@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801051412.m05ECIaH010327@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['stephen.marquard@uct.ac.za']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['stephen.marquard@uct.ac.za']
['louis@media.berkeley.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801042308.m04N8v6O008125@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['zqian@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801042109.m04L92hb007923@nakamura.uits.iupui.edu>']
```

['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['rjlowe@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801042044.m04Kiem3007881@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['rjlowe@iupui.edu']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['rjlowe@iupui.edu']
['zqian@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801042001.m04K1cO0007738@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['zqian@umich.edu']
['rjlowe@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041948.m04JmdwO007705@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['rjlowe@iupui.edu']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['rjlowe@iupui.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041635.m04GZQGZ007313@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['hu2@iupui.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041633.m04GX6eG007292@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['hu2@iupui.edu']

['gsilver@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041611.m04GB1Lb007221@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['gsilver@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041610.m04GA5KP007209@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['zqian@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041609.m04G9EuX007197@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['gsilver@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041608.m04G8d7w007184@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['wagnermr@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041537.m04Fb6Ci007092@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['wagnermr@iupui.edu']
['source@collab.sakaiproject.org']
['wagnermr@iupui.edu']
['wagnermr@iupui.edu']
['zqian@umich.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801041515.m04FFv42007050@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']

['antranig@caret.cam.ac.uk']
['<postmaster@collab.sakaiproject.org>']
['<200801041502.m04F21Jo007031@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['antranig@caret.cam.ac.uk']
['source@collab.sakaiproject.org']
['antranig@caret.cam.ac.uk']
['antranig@caret.cam.ac.uk']
['gopal.ramasammycook@gmail.com']
['<postmaster@collab.sakaiproject.org>']
['<200801041403.m04E3psW006926@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['gopal.ramasammycook@gmail.com']
['source@collab.sakaiproject.org']
['gopal.ramasammycook@gmail.com']
['gopal.ramasammycook@gmail.com']
['david.horwitz@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801041200.m04C0gfK006793@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x>']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x>']
['david.horwitz@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801041106.m04B6lK3006677@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801040947.m049lUxo006517@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['josrodri@iupui.edu']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x>']
['david.horwitz@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801040932.m049W2i5006493@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']

['source@collab.sakaiproject.org;']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['josrodri@iupui.edu']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x>']
['stephen.marquard@uct.ac.za']
['<postmaster@collab.sakaiproject.org>']
['<200801040905.m0495rWB006420@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['stephen.marquard@uct.ac.za']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['stephen.marquard@uct.ac.za']
['louis@media.berkeley.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801040023.m040NpCc005473@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032216.m03MGhDa005292@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['ray@media.berkeley.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032205.m03M5Ea7005273@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['ray@media.berkeley.edu']
['source@collab.sakaiproject.org']
['ray@media.berkeley.edu']
['ray@media.berkeley.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032133.m03LX3gG005191@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032127.m03LRUqH005177@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']

['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['wagnermr@iupui.edu']
['cwen@iupui.edu']
['<postmaster@collab.sakaiproject.org>']
['<200801032122.m03LMFo4005148@nakamura.uits.iupui.edu>']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['<source@collab.sakaiproject.org>;']
['apache@localhost)']
['source@collab.sakaiproject.org;']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['wagnermr@iupui.edu']

6.Search for lines that have an at sign between characters. The characters must be a letter or number and print them

In [20]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('[a-zA-Z0-9]\S+@\S+[a-zA-Z]', line)
    if len(x) > 0:
        print(x)
```

['stephen.marquard@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801051412.m05ECIaH010327@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['stephen.marquard@uct.ac.za']
['louis@media.berkeley.edu']
['postmaster@collab.sakaiproject.org']
['200801042308.m04N8v6O008125@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['zqian@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801042109.m04L92hb007923@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']

['zqian@umich.edu']
['rjlowe@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801042044.m04Kiem3007881@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['rjlowe@iupui.edu']
['zqian@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801042001.m04K1cO0007738@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['zqian@umich.edu']
['rjlowe@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801041948.m04JmdwO007705@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['source@collab.sakaiproject.org']
['rjlowe@iupui.edu']
['rjlowe@iupui.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801041635.m04GZQGZ007313@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['hu2@iupui.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801041633.m04GX6eG007292@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['hu2@iupui.edu']
['gsilver@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041611.m04GB1Lb007221@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']

['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['gsilver@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041610.m04GA5KP007209@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['zqian@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041609.m04G9EuX007197@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['gsilver@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041608.m04G8d7w007184@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['source@collab.sakaiproject.org']
['gsilver@umich.edu']
['gsilver@umich.edu']
['wagnermr@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801041537.m04Fb6Ci007092@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['wagnermr@iupui.edu']
['source@collab.sakaiproject.org']
['wagnermr@iupui.edu']
['wagnermr@iupui.edu']
['zqian@umich.edu']
['postmaster@collab.sakaiproject.org']
['200801041515.m04FFv42007050@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['source@collab.sakaiproject.org']
['zqian@umich.edu']
['zqian@umich.edu']
['antranig@caret.cam.ac.uk']
['postmaster@collab.sakaiproject.org']
['200801041502.m04F21Jo007031@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']

['antranig@caret.cam.ac.uk']
['source@collab.sakaiproject.org']
['antranig@caret.cam.ac.uk']
['antranig@caret.cam.ac.uk']
['gopal.ramasammycook@gmail.com']
['postmaster@collab.sakaiproject.org']
['200801041403.m04E3psW006926@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['gopal.ramasammycook@gmail.com']
['source@collab.sakaiproject.org']
['gopal.ramasammycook@gmail.com']
['gopal.ramasammycook@gmail.com']
['david.horwitz@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801041200.m04C0gfK006793@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x']
['david.horwitz@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801041106.m04B6lK3006677@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801040947.m049lUxo006517@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['josrodri@iupui.edu']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x']
['david.horwitz@uct.ac.za']
['postmaster@collab.sakaiproject.org']
['200801040932.m049W2i5006493@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['source@collab.sakaiproject.org']
['david.horwitz@uct.ac.za']
['david.horwitz@uct.ac.za']
['josrodri@iupui.edu']
['dhorwitz@david-horwitz-6:~/branchManagemnt/sakai_2-5-x']
['stephen.marquard@uct.ac.za']

['postmaster@collab.sakaiproject.org']
['200801040905.m0495rWB006420@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['source@collab.sakaiproject.org']
['stephen.marquard@uct.ac.za']
['stephen.marquard@uct.ac.za']
['louis@media.berkeley.edu']
['postmaster@collab.sakaiproject.org']
['200801040023.m040NpCc005473@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['postmaster@collab.sakaiproject.org']
['200801032216.m03MGhDa005292@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['source@collab.sakaiproject.org']
['louis@media.berkeley.edu']
['louis@media.berkeley.edu']
['ray@media.berkeley.edu']
['postmaster@collab.sakaiproject.org']
['200801032205.m03M5Ea7005273@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['ray@media.berkeley.edu']
['source@collab.sakaiproject.org']
['ray@media.berkeley.edu']
['ray@media.berkeley.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801032133.m03LX3gG005191@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801032127.m03LRUqH005177@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['wagnermr@iupui.edu']

['cwen@iupui.edu']
['postmaster@collab.sakaiproject.org']
['200801032122.m03LMFo4005148@nakamura.uits.iupui.edu']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['source@collab.sakaiproject.org']
['apache@localhost']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['source@collab.sakaiproject.org']
['cwen@iupui.edu']
['cwen@iupui.edu']
['wagnermr@iupui.edu']

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.

In [21]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^X\S*: [0-9.]+', line):
        print(line)
```

```
X-DSPAM-Confidence: 0.8475
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6178
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6961
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7565
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7626
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7556
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7002
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7615
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7601
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7605
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6959
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7606
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7559
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7605
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6932
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7558
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6526
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6948
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6528
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7002
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7554
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6956
```

```
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.6959
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.7556
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.9846
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.8509
X-DSPAM-Probability: 0.0000
X-DSPAM-Confidence: 0.9907
X-DSPAM-Probability: 0.0000
```

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.

In [22]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('^X\S*: ([0-9.]+)', line)
    if len(x) > 0:
        print(x)
```

```
['0.8475']
['0.0000']
['0.6178']
['0.0000']
['0.6961']
['0.0000']
['0.7565']
['0.0000']
['0.7626']
['0.0000']
['0.7556']
['0.0000']
['0.7002']
['0.0000']
['0.7615']
['0.0000']
['0.7601']
['0.0000']
['0.7605']
['0.0000']
['0.6959']
['0.0000']
['0.7606']
['0.0000']
['0.7559']
['0.0000']
['0.7605']
['0.0000']
['0.6932']
['0.0000']
['0.7558']
['0.0000']
['0.6526']
['0.0000']
['0.6948']
['0.0000']
['0.6528']
['0.0000']
['0.7002']
['0.0000']
['0.7554']
['0.0000']
['0.6956']
['0.0000']
```

```
['0.6959']
['0.0000']
['0.7556']
['0.0000']
['0.9846']
['0.0000']
['0.8509']
['0.0000']
['0.9907']
['0.0000']
```

9.Search for lines that start with 'Details: rev=', followed by numbers and '.'. Then print the number if it is greater than zero

In [23]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('^Details:.*rev=([0-9.]+)', line)
    if len(x) > 0:
        print(x)
```

```
['39772']
['39771']
['39770']
['39769']
['39766']
['39765']
['39764']
['39763']
['39762']
['39761']
['39760']
['39759']
['39758']
['39757']
['39756']
['39755']
['39754']
['39753']
['39752']
['39751']
['39750']
['39749']
['39746']
['39745']
['39744']
['39743']
['39742']
```

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 [25]:

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    x = re.findall('^From .* ([0-9][0-9]):', line)
    if len(x) > 0: print(x)
```

```
['09']
['18']
['16']
```

['15']
['15']
['14']
['11']
['11']
['11']
['11']
['11']
['11']
['10']
['10']
['10']
['09']
['07']
['06']
['04']
['04']
['04']
['19']
['17']
['17']
['16']
['16']
['16']

Question2. 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 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 Abagail 895 Abbey 695 Abbie 650 ...

In [26]:

```
import re
import sys

def extract_names(filename):
  """
  Given a file name for baby.html, returns a list starting with the year string
  followed by the name-rank strings in alphabetical order.
  ['2006', 'Aaliyah 91', Aaron 57', 'Abagail 895', ' ...]
  """

  names = []
  f = open(filename, 'r')
  text = f.read()
  year_match = re.search(r'Popularity\sin\s(\d\d\d\d)', text)
  if not year_match:
    sys.stderr.write('Couldn\'t find the year!\n')
    sys.exit(1)
  year = year_match.group(1)
  names.append(year)

  tuples = re.findall(r'<td>(\d+)</td><td>(\w+)</td>\<td>(\w+)</td>', text)

  names_to_rank =  {}
  for rank_tuple in tuples:
    (rank, boyname, girlname) = rank_tuple
    if boyname not in names_to_rank:
      names_to_rank[boyname] = rank
    if girlname not in names_to_rank:
      names_to_rank[girlname] = rank

  sorted_names = sorted(names_to_rank.keys())

  for name in sorted_names:
    names.append(name + " " + names_to_rank[name])

  return names
```

In [28]:

```
extract_names('baby2006.html')
```

Out[28]:

```
['2006',
 'Aaliyah 91',
 'Aaron 57',
 'Abagail 895',
 'Abbey 695',
 'Abbie 650',
 'Abbigail 490',
 'Abby 205',
 'Abdullah 888',
 'Abel 338',
 'Abigail 6',
 'Abigale 889',
 'Abigayle 777',
 'Abraham 183',
 'Abram 586',
 'Abril 845',
 'Ace 838',
 'Ada 715',
 'Adam 64',
 'Adamaris 920',
 'Adan 302',
 'Addison 27',
 'Addisyn 612',
 'Addyson 370',
 'Adelaide 921',
 'Adeline 467',
 'Aden 244',
 'Adison 767',
 'Aditya 810',
```

'Adolfo 581',
'Adonis 822',
'Adrian 63',
'Adriana 106',
'Adrianna 179',
'Adriel 707',
'Adrien 754',
'Adrienne 736',
'Adyson 877',
'Aedan 760',
'Agustin 657',
'Ahmad 524',
'Ahmed 559',
'Aidan 44',
'Aiden 30',
'Aidyn 910',
'Aileen 537',
'Aimee 763',
'Ainsley 456',
'Aisha 683',
'Aiyana 739',
'Akeelah 711',
'Akira 965',
'Alaina 248',
'Alan 122',
'Alana 166',
'Alani 933',
'Alanna 473',
'Alayna 268',
'Albert 354',
'Alberto 315',
'Alden 811',
'Aldo 587',
'Aleah 667',
'Alec 328',
'Aleena 830',
'Alejandra 232',
'Alejandro 96',
'Alena 665',
'Alessandra 361',
'Alessandro 608',
'Alex 67',
'Alexa 39',
'Alexander 12',
'Alexandra 40',
'Alexandria 147',
'Alexandro 861',
'Alexia 164',
'Alexis 14',
'Alexus 545',
'Alexzander 615',
'Alfonso 552',
'Alfred 755',
'Alfredo 334',
'Ali 360',
'Alia 928',
'Alice 383',
'Alicia 167',
'Alijah 584',
'Alina 355',
'Alisa 890',
'Alisha 584',
'Alison 271',
'Alissa 377',
'Alivia 254',
'Aliya 699',
'Aliyah 236',
'Aliza 856',
'Alize 959',
'Allan 507',
'Allen 283',
'Allie 229',
'Allison 46',

'Allison 46',
'Ally 630',
'Allyson 259',
'Alma 616',
'Alondra 170',
'Alonso 643',
'Alonzo 558',
'Alvaro 570',
'Alvin 513',
'Alyson 471',
'Alyssa 19',
'Amanda 102',
'Amani 687',
'Amara 573',
'Amare 778',
'Amari 421',
'Amaris 909',
'Amaya 215',
'Amber 136',
'Amelia 82',
'Amelie 784',
'America 458',
'Amina 978',
'Amir 324',
'Amira 571',
'Amirah 971',
'Amiya 742',
'Amiyah 635',
'Amy 128',
'Amya 476',
'Ana 143',
'Anabelle 757',
'Anahi 287',
'Anais 869',
'Anastasia 288',
'Anaya 486',
'Anders 987',
'Anderson 399',
'Andre 207',
'Andrea 59',
'Andreas 880',
'Andres 161',
'Andrew 8',
'Andy 204',
'Angel 31',
'Angela 114',
'Angelica 208',
'Angelina 48',
'Angeline 863',
'Angelique 693',
'Angelo 262',
'Angie 389',
'Anika 485',
'Aniya 262',
'Aniyah 220',
'Ann 731',
'Anna 23',
'Annabel 848',
'Annabella 556',
'Annabelle 206',
'Annalise 589',
'Anne 460',
'Annette 881',
'Annie 398',
'Annika 335',
'Ansley 704',
'Anthony 9',
'Antoine 631',
'Anton 725',
'Antonio 93',
'Antony 815',
'Antwan 828',

'Anya 405',
'April 319',
'Arabella 653',
'Araceli 570',
'Aracely 722',
'Areli 979',
'Arely 576',
'Ari 634',
'Aria 661',
'Ariana 78',
'Arianna 77',
'Ariel 202',
'Arielle 587',
'Arjun 720',
'Armando 264',
'Armani 698',
'Arnav 932',
'Aron 640',
'Arthur 377',
'Arturo 323',
'Aryan 684',
'Aryana 823',
'Aryanna 652',
'Asa 623',
'Ashanti 882',
'Asher 252',
'Ashlee 338',
'Ashleigh 688',
'Ashley 12',
'Ashly 780',
'Ashlyn 140',
'Ashlynn 293',
'Ashton 121',
'Asia 332',
'Aspen 572',
'Athena 504',
'Atticus 767',
'Aubree 426',
'Aubrey 92',
'Aubrie 604',
'Audrey 68',
'August 618',
'Augustus 831',
'Aurora 312',
'Austen 924',
'Austin 41',
'Austyn 974',
'Autumn 95',
'Ava 5',
'Averie 899',
'Avery 52',
'Axel 295',
'Ayana 700',
'Ayanna 527',
'Aydan 576',
'Ayden 119',
'Aydin 731',
'Ayla 264',
'Aylin 776',
'Bailee 651',
'Bailey 112',
'Barbara 561',
'Barrett 762',
'Baylee 469',
'Beatrice 966',
'Beau 438',
'Beckett 758',
'Belen 914',
'Belinda 809',
'Bella 181',
'Ben 555',
'Benjamin 24',

'Bennett 369',
'Benny 962',
'Bernard 945',
'Bernardo 975',
'Bethany 244',
'Bethzy 878',
'Betsy 743',
'Bianca 182',
'Billy 473',
'Blaine 572',
'Blaise 985',
'Blake 97',
'Blanca 800',
'Blaze 868',
'Bo 771',
'Bobby 480',
'Bode 848',
'Boston 626',
'Brad 897',
'Braden 141',
'Bradley 188',
'Brady 105',
'Bradyn 675',
'Braeden 331',
'Braedon 744',
'Braelyn 855',
'Braiden 602',
'Branden 472',
'Brandi 967',
'Brandon 27',
'Brandy 801',
'Branson 862',
'Braulio 966',
'Braxton 224',
'Brayan 294',
'Brayden 79',
'Braydon 396',
'Braylen 756',
'Braylon 401',
'Breana 844',
'Breanna 126',
'Bree 955',
'Brenda 239',
'Brendan 185',
'Brenden 317',
'Brendon 501',
'Brenna 381',
'Brennan 299',
'Brennen 732',
'Brent 481',
'Brenton 942',
'Brett 304',
'Bria 870',
'Brian 72',
'Briana 121',
'Brianna 20',
'Brice 695',
'Bridget 347',
'Brielle 459',
'Briley 960',
'Brisa 605',
'Britney 474',
'Brittany 318',
'Brittney 626',
'Brock 261',
'Broderick 884',
'Brodie 442',
'Brody 147',
'Brogan 999',
'Brooke 44',
'Brooklyn 67',
'Brooklynn 237',

'Brooks 592',
'Bruce 482',
'Bruno 793',
'Bryan 66',
'Bryanna 424',
'Bryant 389',
'Bryce 109',
'Brycen 588',
'Brylee 885',
'Brynn 402',
'Bryson 176',
'Byron 523',
'Cade 288',
'Caden 91',
'Cadence 214',
'Cael 917',
'Caiden 312',
'Cailyn 872',
'Caitlin 207',
'Caitlyn 199',
'Cale 733',
'Caleb 34',
'Cali 602',
'Callie 303',
'Calvin 220',
'Camden 221',
'Cameron 52',
'Camila 180',
'Camilla 825',
'Camille 308',
'Campbell 659',
'Camren 579',
'Camron 345',
'Camryn 216',
'Cannon 796',
'Cara 559',
'Carina 864',
'Carissa 585',
'Carl 429',
'Carla 501',
'Carlee 654',
'Carley 550',
'Carlie 592',
'Carlo 937',
'Carlos 70',
'Carly 251',
'Carmelo 870',
'Carmen 258',
'Carmine 785',
'Carol 968',
'Carolina 281',
'Caroline 89',
'Carolyn 517',
'Carrie 859',
'Carson 87',
'Carter 75',
'Case 951',
'Casey 308',
'Cash 378',
'Cason 753',
'Cassandra 219',
'Cassidy 198',
'Cassie 732',
'Catalina 696',
'Catherine 122',
'Cayden 213',
'Cayla 886',
'Cecelia 669',
'Cecilia 265',
'Cedric 595',
'Celeste 327',
'Celia 707',

'Cesar 168',
'Chad 375',
'Chaim 946',
'Chana 990',
'Chance 273',
'Chandler 402',
'Chanel 917',
'Charity 673',
'Charlee 915',
'Charles 60',
'Charlie 337',
'Charlize 751',
'Charlotte 123',
'Chase 83',
'Chasity 744',
'Chaya 785',
'Chaz 905',
'Chelsea 192',
'Chelsey 790',
'Cherish 754',
'Cheyanne 657',
'Cheyenne 171',
'Chloe 18',
'Chris 358',
'Christian 21',
'Christiana 984',
'Christina 158',
'Christine 437',
'Christopher 7',
'Ciara 213',
'Cierra 563',
'Cindy 365',
'Citlali 948',
'Claire 86',
'Clara 233',
'Clare 663',
'Clarence 818',
'Clarissa 468',
'Clark 696',
'Claudia 339',
'Clay 708',
'Clayton 226',
'Clinton 844',
'Cloe 941',
'Coby 832',
'Cody 106',
'Coen 963',
'Cohen 415',
'Colby 271',
'Cole 84',
'Coleman 661',
'Colin 111',
'Colleen 902',
'Collin 181',
'Colt 906',
'Colten 510',
'Colton 133',
'Conner 144',
'Connor 53',
'Conor 496',
'Conrad 788',
'Cooper 113',
'Cora 384',
'Corbin 289',
'Corey 234',
'Corinne 826',
'Cornelius 939',
'Cortez 938',
'Cory 431',
'Courtney 190',
'Craig 548',
'Cristal 660',

'Cristian 132',
'Cristina 495',
'Cristobal 991',
'Cristofer 804',
'Cristopher 499',
'Cruz 500',
'Crystal 201',
'Cullen 790',
'Curtis 339',
'Cynthia 240',
'Cyrus 515',
'Dahlia 988',
'Daisy 149',
'Dakota 172',
'Dale 745',
'Dalia 849',
'Dallas 352',
'Dalton 199',
'Damarion 646',
'Damaris 609',
'Damian 136',
'Damien 196',
'Damion 497',
'Damon 386',
'Dana 395',
'Dandre 992',
'Dane 393',
'Dangelo 863',
'Dania 985',
'Danica 352',
'Daniel 6',
'Daniela 132',
'Daniella 307',
'Danielle 116',
'Danika 569',
'Danna 518',
'Danny 307',
'Dante 291',
'Daphne 606',
'Darian 590',
'Darien 830',
'Dario 875',
'Darion 869',
'Darius 280',
'Darnell 711',
'Darrell 597',
'Darren 361',
'Darrius 930',
'Darryl 773',
'Darwin 772',
'Dashawn 690',
'Davian 682',
'David 13',
'Davin 601',
'Davion 478',
'Davis 405',
'Davon 603',
'Dawson 233',
'Dayana 553',
'Dayanara 435',
'Dayton 540',
'Deacon 687',
'Dean 385',
'Deandre 452',
'Deangelo 794',
'Deanna 532',
'Deborah 676',
'Declan 364',
'Deja 753',
'Delaney 193',
'Delilah 548',
'Demarcus 735',

'Demarion 749',
'Demetrius 437',
'Denise 379',
'Dennis 313',
'Denzel 940',
'Deon 849',
'Derek 159',
'Derick 736',
'Derrick 256',
'Deshaun 853',
'Deshawn 518',
'Desirae 883',
'Desiree 300',
'Desmond 464',
'Destin 960',
'Destinee 554',
'Destiney 972',
'Destini 929',
'Destiny 37',
'Devan 582',
'Deven 606',
'Devin 100',
'Devon 197',
'Devonte 841',
'Devyn 748',
'Dexter 913',
'Diamond 316',
'Diana 120',
'Diego 56',
'Dillan 774',
'Dillon 223',
'Dion 988',
'Domenic 964',
'Dominic 85',
'Dominick 216',
'Dominik 612',
'Dominique 648',
'Donald 303',
'Donavan 812',
'Donna 832',
'Donovan 198',
'Donte 722',
'Dorian 469',
'Douglas 365',
'Drake 239',
'Draven 685',
'Drew 205',
'Dulce 274',
'Duncan 654',
'Dustin 259',
'Dwayne 610',
'Dylan 26',
'Ean 775',
'Earl 993',
'Easton 359',
'Eddie 395',
'Eddy 855',
'Eden 320',
'Edgar 171',
'Edison 947',
'Edith 638',
'Eduardo 126',
'Edward 143',
'Edwin 158',
'Efrain 663',
'Efren 968',
'Eileen 770',
'Elaina 448',
'Elaine 719',
'Eleanor 277',
'Elena 187',
'Eli 139',

'Eliana 282',
'Elias 186',
'Elijah 29',
'Elisa 543',
'Elisabeth 502',
'Elise 218',
'Elisha 629',
'Eliza 325',
'Elizabeth 11',
'Ella 21',
'Elle 480',
'Ellen 544',
'Elliana 857',
'Ellie 175',
'Elliot 372',
'Elliott 388',
'Ellis 783',
'Elmer 907',
'Elsa 792',
'Elsie 879',
'Elvis 761',
'Elyse 684',
'Emanuel 263',
'Emely 309',
'Emerson 305',
'Emery 778',
'Emilee 375',
'Emilia 420',
'Emiliano 332',
'Emilie 491',
'Emilio 298',
'Emily 1',
'Emma 2',
'Emmalee 685',
'Emmanuel 166',
'Emmett 569',
'Enrique 281',
'Enzo 737',
'Eric 77',
'Erica 222',
'Erick 174',
'Ericka 993',
'Erik 189',
'Erika 238',
'Erin 130',
'Ernest 723',
'Ernesto 379',
'Esmeralda 224',
'Esperanza 675',
'Essence 839',
'Esteban 333',
'Estefani 905',
'Estevan 846',
'Esther 298',
'Estrella 311',
'Ethan 4',
'Ethen 952',
'Eugene 647',
'Eva 124',
'Evan 42',
'Evangeline 597',
'Eve 590',
'Evelin 674',
'Evelyn 65',
'Everett 451',
'Ezekiel 269',
'Ezequiel 541',
'Ezra 340',
'Fabian 272',
'Fabiola 980',
'Faith 64',
'Fatima 228',
'Felicity 617',

'Felicity 617',
'Felipe 457',
'Felix 397',
'Fernanda 431',
'Fernando 151',
'Finley 886',
'Finn 456',
'Finnegan 779',
'Fiona 333',
'Flor 1000',
'Frances 779',
'Francesca 428',
'Francis 561',
'Francisco 157',
'Franco 918',
'Frank 245',
'Frankie 648',
'Franklin 439',
'Freddy 641',
'Frederick 483',
'Fredrick 903',
'Frida 836',
'Gabriel 28',
'Gabriela 110',
'Gabriella 50',
'Gabrielle 62',
'Gael 314',
'Gage 156',
'Gaige 919',
'Galilea 840',
'Gannon 889',
'Garret 750',
'Garrett 138',
'Garrison 895',
'Gary 350',
'Gauge 881',
'Gaven 948',
'Gavin 38',
'Gavyn 710',
'Genesis 169',
'Genevieve 368',
'George 153',
'Georgia 273',
'Gerald 544',
'Gerardo 268',
'German 839',
'Gia 643',
'Giana 708',
'Giancarlo 716',
'Gianna 98',
'Gianni 611',
'Gideon 591',
'Gilbert 658',
'Gilberto 554',
'Gillian 758',
'Gina 712',
'Giovani 791',
'Giovanna 723',
'Giovanni 146',
'Giovanny 702',
'Giselle 168',
'Gisselle 677',
'Glenn 850',
'Gloria 453',
'Gonzalo 925',
'Gordon 900',
'Grace 17',
'Gracelyn 759',
'Gracie 103',
'Grady 475',
'Graham 430',
'Grant 155',

'Grayson 218',
'Gregory 208',
'Greta 680',
'Gretchen 771',
'Greyson 503',
'Griffin 254',
'Guadalupe 255',
'Guillermo 440',
'Gunnar 502',
'Gunner 578',
'Gustavo 305',
'Guy 989',
'Gwendolyn 631',
'Haden 858',
'Hadley 494',
'Haiden 872',
'Hailee 454',
'Hailey 25',
'Hailie 760',
'Haleigh 596',
'Haley 75',
'Halie 937',
'Halle 493',
'Hallie 450',
'Hamza 738',
'Hana 716',
'Hanna 269',
'Hannah 8',
'Harley 388',
'Harmony 342',
'Harold 652',
'Harper 510',
'Harrison 232',
'Harry 593',
'Hassan 765',
'Haven 610',
'Hayden 73',
'Haylee 247',
'Hayley 306',
'Haylie 427',
'Hazel 465',
'Heath 786',
'Heather 341',
'Heaven 253',
'Hector 175',
'Heidi 295',
'Helen 343',
'Helena 508',
'Henry 95',
'Hezekiah 885',
'Hillary 982',
'Holden 384',
'Holly 346',
'Hope 200',
'Houston 837',
'Howard 836',
'Hudson 249',
'Hugh 994',
'Hugo 371',
'Humberto 697',
'Hunter 54',
'Ian 81',
'Ibrahim 594',
'Ignacio 703',
'Iliana 727',
'Imani 409',
'Imanol 969',
'India 568',
'Ingrid 619',
'Irene 593',
'Iris 369',
'Irvin 670',

'Isaac 48',
'Isabel 87',
'Isabela 470',
'Isabell 724',
'Isabella 4',
'Isabelle 85',
'Isai 706',
'Isaiah 40',
'Isaias 531',
'Isiah 406',
'Isis 566',
'Ismael 327',
'Israel 203',
'Issac 409',
'Itzel 378',
'Ivan 127',
'Ivy 334',
'Iyana 880',
'Iyanna 991',
'Izabella 290',
'Izabelle 998',
'Izaiah 461',
'Izayah 976',
'Jabari 609',
'Jace 187',
'Jacey 747',
'Jack 35',
'Jackson 36',
'Jaclyn 860',
'Jacob 1',
'Jacoby 909',
'Jacqueline 118',
'Jacquelyn 668',
'Jada 93',
'Jade 111',
'Jaden 88',
'Jadon 398',
'Jadyn 286',
'Jaeden 666',
'Jaelyn 442',
'Jaelynn 793',
'Jaheim 977',
'Jaida 560',
'Jaiden 214',
'Jaidyn 586',
'Jaime 279',
'Jair 726',
'Jairo 564',
'Jakayla 730',
'Jake 107',
'Jakob 293',
'Jalen 228',
'Jaliyah 762',
'Jalyn 956',
'Jalynn 949',
'Jamal 504',
'Jamar 649',
'Jamarcus 914',
'Jamari 444',
'Jamarion 459',
'Jamel 970',
'James 16',
'Jameson 424',
'Jamie 242',
'Jamir 768',
'Jamison 534',
'Jamya 737',
'Jan 751',
'Janae 679',
'Jane 478',
'Janelle 390',
'Janessa 594',

'Janet 562',
'Janiah 824',
'Janice 994',
'Janiya 386',
'Janiyah 461',
'Jaquan 656',
'Jaqueline 512',
'Jared 137',
'Jaron 819',
'Jarrett 659',
'Jarvis 1000',
'Jase 565',
'Jasiah 933',
'Jasmin 183',
'Jasmine 29',
'Jasmyn 926',
'Jason 55',
'Jasper 568',
'Javen 954',
'Javier 162',
'Javion 644',
'Javon 410',
'Jax 995',
'Jaxon 211',
'Jaxson 391',
'Jay 351',
'Jayce 427',
'Jaycee 765',
'Jayda 294',
'Jayden 50',
'Jaydin 971',
'Jaydon 416',
'Jayla 99',
'Jaylah 903',
'Jaylan 692',
'Jaylee 818',
'Jayleen 833',
'Jaylen 191',
'Jaylene 912',
'Jaylin 441',
'Jaylon 479',
'Jaylyn 796',
'Jaylynn 632',
'Jayson 353',
'Jazlyn 533',
'Jazmin 156',
'Jazmine 226',
'Jazmyn 620',
'Jean 721',
'Jefferson 642',
'Jeffery 432',
'Jeffrey 180',
'Jenna 88',
'Jennifer 51',
'Jenny 475',
'Jeramiah 926',
'Jeremiah 71',
'Jeremy 123',
'Jerimiah 931',
'Jermaine 474',
'Jerome 577',
'Jerry 318',
'Jesse 102',
'Jessica 32',
'Jessie 485',
'Jesus 74',
'Jett 535',
'Jewel 861',
'Jillian 174',
'Jimena 472',
'Jimmy 325',
'Joan 978',

```
 'Joana 999',
 'Joanna 256',
 'Joaquin 286',
 'Jocelyn 73',
 'Joe 370',
 'Joel 124',
 'Joey 537',
 'Johan 528',
 'Johana 850',
 'Johanna 376',
 'John 20',
 'Johnathan 177',
 'Johnathon 542',
 'Johnny 237',
 'Johnpaul 979',
 'Jolie 614',
 'Jon 455',
 'Jonah 170',
 'Jonas 357',
 'Jonathan 22',
 'Jonathon 376',
 'Jordan 46',
 'Jorden 759',
 'Jordon 739',
 'Jordy 677',
 'Jordyn 178',
 'Jorge 120',
 'Jorja 969',
 'Jose 32',
 'Josef 972',
 'Joselin 892',
 'Joselyn 314',
 'Joseph 11',
 'Josephine 221',
 'Josh 714',
 ...]
```

In [ ]:

```
```

Lab10. Implementation of Map, Filter and Reduce Function

Question1. Write a program to implement MAP function. Find the square root of a list of numbers [1, 2, 4, 6] using map and sqrt functions. Check the answer against your user defined function mymap().

In [9]:

```
from math import sqrt
lst=[1,2,4,6]
map_object=map(sqrt,lst)
print(list(map_object))


def mymap(f, seq):
    result = []
    for elt in seq:
        result.append(f(elt))
    return result
mymap(sqrt,lst)
```

```
[1.0, 1.4142135623730951, 2.0, 2.449489742783178]
```

Out[9]:

```
[1.0, 1.4142135623730951, 2.0, 2.449489742783178]
```

Question2. Write a program to implement FILTER function. Filter all upper case letters in a list ['x', 'Y', '2', '3', 'Z', 'b'] using filter function. Check the answer against your user define function myfilter().

In [16]:

```
filter_object=filter(str.isupper,['x', 'Y', '2', '3', 'Z', 'b'])
print(list(filter_object))

def filter(f, seq):
    result = []
    for elt in seq:
        if f(elt):
            result.append(elt)
    return result
filter(str.isupper,['x', 'Y', '2', '3', 'Z', 'b'])
```

```
['Y', 'Z']
```

Out[16]:

```
['Y', 'Z']
```

Question3.Write a program to create a lambda function that takes two characters and concatenates them. Now, apply this function inside REDUCE function that will reduce the list of characters ['a', 'b', 'c', d'] with the intial value 'x'.

In [26]:

```
from functools import reduce
(lambda x,y:x+y)('a','b')
reduce (lambda x,y:x+y,['a','b','c','d'],'x')
```

Out[26]:

```
'xabcd'
```

Question4. Imagine an accounting routine used in a book shop. It works on a list with sublists, which look like this:

Write a Python program, which returns a list with 2-tuples. Each tuple consists of an order number and the product of the price per items and the quantity. The product should be decreased by RS 10 if the value of the order is smaller than RS 100.00. Write a Python program using lambda and map functions.

In [1]:

```
tab = [[34587,'Learning Python , Mark Lutz',4,40.95],
       [98762,'Programming Python , Mark Lutz',5,56.80],
       [77226,'Head first Python , Paul Barry',3,32.95],
       [88112,'Einfuhrung in Python , Bernd Klein',3,24.99]]
print(*tab, sep='\n')
```

```
[34587, 'Learning Python , Mark Lutz', 4, 40.95]
[98762, 'Programming Python , Mark Lutz', 5, 56.8]
[77226, 'Head first Python , Paul Barry', 3, 32.95]
[88112, 'Einfuhrung in Python , Bernd Klein', 3, 24.99]
```

In [2]:

```
prg = []
prg1 = []
final = []
nprg = []
for i in range(4):
    j=tab[i][2]*tab[i][3]
    if j < 100:
        j = j - 10
        nprg.append(round(j,2))
    else:
        nprg.append(round(j,2))
    prg1.append(tab[i][0])
t1=tuple(prg1)
t2=tuple(nprg)
final.append(t1)
final.append(t2)
print(final)
```

```
[(34587, 98762, 77226, 88112), (163.8, 284.0, 88.85, 64.97)]
```

In [ ]:

```
```

Lab11. Retrieving Data From Web and Parsing

1.Retrieve data from web page using URLLIB and print the frequency of words from that page.

In [8]:

```
import urllib.request
counts = dict()
fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhand:
    words = line.decode().split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1
print(counts)
```

```
{'But': 1, 'soft': 1, 'what': 1, 'light': 1, 'through': 1, 'yonder': 1, 'window': 1, 'bre
aks': 1, 'It': 1, 'is': 3, 'the': 3, 'east': 1, 'and': 3, 'Juliet': 1, 'sun': 2, 'Arise':
1, 'fair': 1, 'kill': 1, 'envious': 1, 'moon': 1, 'Who': 1, 'already': 1, 'sick': 1, 'pal
e': 1, 'with': 1, 'grief': 1}
```

2.Retrieve and display all hyperlinks (ie., HREF attribute) from a webpage using BeautifulSoup

In [15]:

```
import urllib.request,urllib.parse,urllib.error
from bs4 import BeautifulSoup
import ssl
ctx=ssl.create_default_context()
ctx.check_hostname=False
ctx.verify_mode = ssl.CERT_NONE
url=input('enter url')
html=urllib.request.urlopen(url,context=ctx).read()
soup=BeautifulSoup(html,'html.parser')
tags=soup('a')
for tag in tags:
    print(tag.get('href',None))
```

```
enter urlhttps://www.crummy.com/software/BeautifulSoup/bs4/doc/
genindex.html
#

#beautiful-soup-documentation
http://www.crummy.com/software/BeautifulSoup/
http://www.crummy.com/software/BeautifulSoup/bs3/documentation.html
#porting-code-to-bs4
https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/
http://kondou.com/BS4/
https://www.crummy.com/software/BeautifulSoup/bs4/doc.ko/
https://www.crummy.com/software/BeautifulSoup/bs4/doc.ptbr
https://www.crummy.com/software/BeautifulSoup/bs4/doc.ru/
#getting-help
https://groups.google.com/forum/?fromgroups#!forum/beautifulsoup
#diagnose
#quick-start
#installing-beautiful-soup
http://www.crummy.com/software/BeautifulSoup/bs3/documentation.html
http://www.crummy.com/software/BeautifulSoup/download/4.x/
#problems-after-installation
#installing-a-parser
http://lxml.de/
http://code.google.com/p/html5lib/
#differences-between-parsers
#making-the-soup
#id17
#kinds-of-objects
#tag
```

#navigating-the-tree
#searching-the-tree
#name
#attributes
#multi-valued-attributes
#navigablestring
#navigating-the-tree
#searching-the-tree
#replace-with
#navigating-the-tree
#searching-the-tree
#beautifulsoup
#tag
#navigating-the-tree
#searching-the-tree
#modifying-the-tree
#tag
#comments-and-other-special-strings
#navigating-the-tree
#going-down
#navigating-using-tag-names
#searching-the-tree
#contents-and-children
#descendants
#string
#strings-and-stripped-strings
#going-up
#parent
#parents
#going-sideways
#next-sibling-and-previous-sibling
#next-siblings-and-previous-siblings
#going-back-and-forth
#next-element-and-previous-element
#next-elements-and-previous-elements
#searching-the-tree
#kinds-of-filters
#a-string
#a-regular-expression
#a-list
#true
#a-function
#find-all
#id12
#attrs
#recursive
#id13
#limit
#kwargs
#kinds-of-filters
#the-name-argument
#kinds-of-filters
#a-string
#a-regular-expression
#a-list
#a-function
#the-value-true
#the-keyword-arguments
#a-string
#a-regular-expression
#a-list
#a-function
#the-value-true
#searching-by-css-class
#multivalue
#the-string-argument
#a-string
#a-regular-expression
#a-list
#a-function
#the-value-true
#the-limit-argument

#the-recursive-argument
#calling-a-tag-is-like-calling-find-all
#find
#id12
#attrs
#recursive
#id13
#kwargs
#navigating-using-tag-names
#find-parents-and-find-parent
#id12
#attrs
#id13
#limit
#kwargs
#id12
#attrs
#id13
#kwargs
#parent
#parents
#find-next-siblings-and-find-next-sibling
#id12
#attrs
#id13
#limit
#kwargs
#id12
#attrs
#id13
#kwargs
#sibling-generators
#find-previous-siblings-and-find-previous-sibling
#id12
#attrs
#id13
#limit
#kwargs
#id12
#attrs
#id13
#kwargs
#sibling-generators
#find-all-next-and-find-next
#id12
#attrs
#id13
#limit
#kwargs
#id12
#attrs
#id13
#kwargs
#element-generators
#find-all-previous-and-find-previous
#id12
#attrs
#id13
#limit
#kwargs
#id12
#attrs
#id13
#kwargs
#element-generators
#css-selectors
https://facelessuser.github.io/soupsieve/
https://facelessuser.github.io/soupsieve/
#modifying-the-tree
#changing-tag-names-and-attributes
#attributes
#modifying-string

#append
#extend
#navigablestring-and-new-tag
#insert
#insert-before-and-insert-after
#clear
#extract
#decompose
#replace-with
#wrap
#unwrap
#smooth
#output
#pretty-printing
#non-pretty-printing
#encodings
#output-formatters
#get-text
#string-generators
#specifying-the-parser-to-use
#installing-a-parser
#differences-between-parsers
#encodings
#unicode-dammit
#output-encoding
#unicode-dammit
#smart-quotes
#inconsistent-encodings
#line-numbers
#comparing-objects-for-equality
#copying-beautiful-soup-objects
#advanced-parser-customization
#parsing-only-part-of-a-document
#soupstrainer
#searching-the-tree
#id12
#attrs
#id13
#kwargs
#searching-the-tree
#customizing-multi-valued-attributes
#handling-duplicate-attributes
#instantiating-custom-subclasses
#troubleshooting
#diagnose
#errors-when-parsing-a-document
#installing-a-parser
#parser-installation
#parser-installation
#version-mismatch-problems
#parsing-xml
#parser-installation
#other-parser-problems
#differences-between-parsers
http://www.w3.org/TR/html5/syntax.html#syntax
#parsing-xml
#miscellaneous
http://wiki.python.org/moin/PrintFails
#improving-performance
http://lxml.de/
#parser-installation
http://pypi.python.org/pypi/cchardet/
#parsing-only-part-of-a-document
#translating-this-documentation
#id18
http://www.crummy.com/software/BeautifulSoup/bs3/download/3.x/BeautifulSoup-3.2.0.tar.gz
http://www.crummy.com/software/BeautifulSoup/bs3/documentation.html
#porting-code-to-bs4
http://www.python.org/dev/peps/pep-0008/
#you-need-a-parser
#installing-a-parser
#method-names

#generators
#string-generators
#xml
#entities
#unicode-dammit
#output-formatters
#id19
#string
#multi-valued-attributes
#id13
#id12
#string
#id13
#
#
#getting-help
#quick-start
#installing-beautiful-soup
#problems-after-installation
#installing-a-parser
#making-the-soup
#kinds-of-objects
#tag
#name
#attributes
#multi-valued-attributes
#navigablestring
#beautifulsoup
#comments-and-other-special-strings
#navigating-the-tree
#going-down
#navigating-using-tag-names
#contents-and-children
#descendants
#string
#strings-and-stripped-strings
#going-up
#parent
#parents
#going-sideways
#next-sibling-and-previous-sibling
#next-siblings-and-previous-siblings
#going-back-and-forth
#next-element-and-previous-element
#next-elements-and-previous-elements
#searching-the-tree
#kinds-of-filters
#a-string
#a-regular-expression
#a-list
#true
#a-function
#find-all
#the-name-argument
#the-keyword-arguments
#searching-by-css-class
#the-string-argument
#the-limit-argument
#the-recursive-argument
#calling-a-tag-is-like-calling-find-all
#find
#find-parents-and-find-parent
#find-next-siblings-and-find-next-sibling
#find-previous-siblings-and-find-previous-sibling
#find-all-next-and-find-next
#find-all-previous-and-find-previous
#css-selectors
#modifying-the-tree
#changing-tag-names-and-attributes
#modifying-string
#append
#extend

```
#navigablestring-and-new-tag
#insert
#insert-before-and-insert-after
#clear
#extract
#decompose
#replace-with
#wrap
#unwrap
#smooth
#output
#pretty-printing
#non-pretty-printing
#output-formatters
#get-text
#specifying-the-parser-to-use
#differences-between-parsers
#encodings
#output-encoding
#unicode-dammit
#smart-quotes
#inconsistent-encodings
#line-numbers
#comparing-objects-for-equality
#copying-beautiful-soup-objects
#advanced-parser-customization
#parsing-only-part-of-a-document
#soupstrainer
#customizing-multi-valued-attributes
#handling-duplicate-attributes
#instantiating-custom-subclasses
#troubleshooting
#diagnose
#errors-when-parsing-a-document
#version-mismatch-problems
#parsing-xml
#other-parser-problems
#miscellaneous
#improving-performance
#translating-this-documentation
#id18
#porting-code-to-bs4
#you-need-a-parser
#method-names
#generators
#xml
#entities
#id19
_sources/index.rst.txt
genindex.html
#

https://www.sphinx-doc.org/
```

3.Create a HTML file for the following Student Marks and print the number of students and their names and marks.

In [2]:

```
from IPython.core.display import HTML

que3 = '''
<table>
    <tr>
        <td>ID</td>
        <td>NAME</td>
        <td>MARK1</td>
        <td>MARK2</td>
        <td>MARK3</td>
    </tr>
    <tr>
```

```
        <td>DS01</td>
        <td>REX</td>
        <td>87</td>
        <td>57</td>
        <td>74</td>
    </tr>
    <tr>
        <td>DS02</td>
        <td>PETER</td>
        <td>68</td>
        <td>98</td>
        <td>55</td>
    </tr>
</table>

'''

HTML(que3)
```

Out[2]:

ID	NAME	MARK1	MARK2	MARK3
DS01	REX	87	57	74
DS02	PETER	68	98	55

4.Create a JSON file for the following Students Marks and print the number of students and their names and marks. ID: DS01: Name: rex Semester1: 80, 55 Semester2: 50, 70, 82 ID: DS02: Name: peter Semester1: 92, 75

In [23]:

```
import json
data = '''
[
{"id" : "ds01",
 "name" : "rex",
 "semester1":"80,55",
 "semester2":"78,98"
},
{"id" : "ds02",
"name" :"Brent",
"semester1":"35,67",
"semester2":"65,87"
}
]'''
info = json.loads(data)
print('User count:', len(info))
for item in info:
        print('Name', item['name'])
        print('Id',item['id'])
        print('semester1', item['semester1'])
        print('semester2',item['semester2'])
```

```
User count: 2
Name rex
Id ds01
semester1 80,55
semester2 78,98
Name Brent
Id ds02
semester1 35,67
semester2 65,87
```

5.Crawl Weather of a City and Display 7 Day Forecast. Find weather data of Tiruchirappalli city from some website such as www.weather.com Exploring page structure with Chrome DevTools Extract information from web page and display the weather forecast for 10 days or 7 days Reference:
https://www.dataquest.io/blog/web-scraping-tutorial-python/

In [1]:

```
import requests
from bs4 import BeautifulSoup

page=requests.get("https://weather.com/en-IN/weather/tenday/l/d251a574fa1fa7a38aef6a630cb
7e91a5c92cc2165acc34a8ba2f7e589c0f61e")
page.content

bs4=BeautifulSoup(page.content,'html.parser')
bs4.find_all('p')
(bs4.find_all(class_='DetailsSummary--DetailsSummary--QpFD-'))
days=[bs4.find_all('h2')[day].get_text() for day in range(len(bs4.find_all('h2')))]
tempr=[bs4.find_all(class_="DetailsSummary--temperature--3FMlw")[temp].get_text() for tem
p in range(len(bs4.find_all(class_="DetailsSummary--temperature--3FMlw")))]


days=days[1:12]
tempr=tempr[1:12]

from IPython.display import display
import pandas as pd
#creating a data frame to store the values
b={"Days": days,
   "Temperature": tempr}
weather = pd.DataFrame(b)

weather
```

Out[1]:

	Days	Temperature
0	Tonight	34°/22°
1	Sun 14	35°/22°
2	Mon 15	35°/21°
3	Tue 16	35°/22°
4	Wed 17	34°/22°
5	Thu 18	33°/22°
6	Fri 19	33°/22°
7	Sat 20	32°/22°
8	Sun 21	34°/21°
9	Mon 22	35°/21°
10	Tue 23	36°/21°

6.Real Time Stock Prices Crawling and Display of a specified Company Crawl prices of a stock such as InfoSys (Stock Code INFY) for a period of month or year. Plot a line graph of monthly or yearly price movements Crawl prices of one more stock such as CTS Update your line graph with the prices movements of two stocks
Reference: https://ntguardian.wordpress.com/2018/07/17/stock-data-analysis-python-v2/
https://github.com/rajkumarbhc/datascienceutils/blob/master/data_science_utils/financial/Web%20Scraping%20Tutorial.ipynb

In [8]:

```
!pip install yfinance --upgrade --no-cache-dir
import yfinance as yf
yf.pdr_override()
df_info = pdr.get_data_yahoo("INFY", start="2018-01-01").reset_index()
df_info.to_csv('INFY.csv',index=False)
df_info.head()
```

```
Requirement already satisfied: yfinance in c:\windows\system32\src\yfinance (0.1.55)
Requirement already satisfied: pandas>=0.24 in c:\users\mahesh\anaconda3\lib\site-package
s (from yfinance) (1.2.1)
Requirement already satisfied: numpy>=1.15 in c:\users\mahesh\anaconda3\lib\site-packages
```

```
Requirement already satisfied: numpy>=1.15 in c:\users\mahesh\anaconda3\lib\site-packages
(from yfinance) (1.19.2)
Requirement already satisfied: requests>=2.20 in c:\users\mahesh\anaconda3\lib\site-packa
ges (from yfinance) (2.25.1)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\mahesh\anaconda3\lib\site-
packages (from yfinance) (0.0.9)
Requirement already satisfied: lxml>=4.5.1 in c:\users\mahesh\anaconda3\lib\site-packages
(from yfinance) (4.6.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\mahesh\anaconda3\lib\si
te-packages (from pandas>=0.24->yfinance) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\mahesh\anaconda3\lib\site-package
s (from pandas>=0.24->yfinance) (2021.1)
Requirement already satisfied: six>=1.5 in c:\users\mahesh\anaconda3\lib\site-packages (f
rom python-dateutil>=2.7.3->pandas>=0.24->yfinance) (1.15.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\mahesh\anaconda3\lib\sit
e-packages (from requests>=2.20->yfinance) (1.26.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mahesh\anaconda3\lib\site-p
ackages (from requests>=2.20->yfinance) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in c:\users\mahesh\anaconda3\lib\site-package
s (from requests>=2.20->yfinance) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\mahesh\anaconda3\lib\site-pa
ckages (from requests>=2.20->yfinance) (4.0.0)
[**********************100%**********************]  1 of 1 completed
```

Out[8]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-01-02	8.135	8.195	8.115	8.145	7.429263	12298200
1	2018-01-03	8.120	8.135	8.050	8.075	7.365414	10250800
2	2018-01-04	8.100	8.100	8.010	8.025	7.319807	16272000
3	2018-01-05	8.085	8.190	8.075	8.175	7.456626	9813600
4	2018-01-08	8.190	8.260	8.170	8.240	7.515914	11198200

In [7]:

```
!pip install yfinance --upgrade --no-cache-dir
import yfinance as yf
yf.pdr_override()
df_cts = pdr.get_data_yahoo("CTS", start="2018-01-01").reset_index()
df_cts.to_csv('CTS.csv',index=False)
df_cts.head()
```

```
Requirement already satisfied: yfinance in c:\windows\system32\src\yfinance (0.1.55)
Requirement already satisfied: pandas>=0.24 in c:\users\mahesh\anaconda3\lib\site-package
s (from yfinance) (1.2.1)
Requirement already satisfied: numpy>=1.15 in c:\users\mahesh\anaconda3\lib\site-packages
(from yfinance) (1.19.2)
Requirement already satisfied: requests>=2.20 in c:\users\mahesh\anaconda3\lib\site-packa
ges (from yfinance) (2.25.1)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\mahesh\anaconda3\lib\site-
packages (from yfinance) (0.0.9)
Requirement already satisfied: lxml>=4.5.1 in c:\users\mahesh\anaconda3\lib\site-packages
(from yfinance) (4.6.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\mahesh\anaconda3\lib\site-package
s (from pandas>=0.24->yfinance) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\mahesh\anaconda3\lib\si
te-packages (from pandas>=0.24->yfinance) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\mahesh\anaconda3\lib\site-packages (f
rom python-dateutil>=2.7.3->pandas>=0.24->yfinance) (1.15.0)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\mahesh\anaconda3\lib\site-pa
ckages (from requests>=2.20->yfinance) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mahesh\anaconda3\lib\site-p
ackages (from requests>=2.20->yfinance) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\mahesh\anaconda3\lib\sit
e-packages (from requests>=2.20->yfinance) (1.26.3)
Requirement already satisfied: idna<3,>=2.5 in c:\users\mahesh\anaconda3\lib\site-package
s (from requests>=2.20->yfinance) (2.10)
[**********************100%**********************]  1 of 1 completed
```

Out[7]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-01-02	25.900000	26.950001	25.850000	26.500000	26.042004	117900
1	2018-01-03	26.549999	26.700001	26.200001	26.250000	25.796324	62600
2	2018-01-04	26.500000	27.150000	26.450001	26.750000	26.287683	55800
3	2018-01-05	26.750000	27.049999	26.600000	26.850000	26.385950	39700
4	2018-01-08	26.750000	26.900000	26.549999	26.700001	26.238546	40100

In [9]:

```
import matplotlib.pyplot as plt
%matplotlib inline

df_info["Adj Close"].plot(grid = True)
```

Out[9]:

<AxesSubplot:>

In [10]:

```
import matplotlib.pyplot as plt
%matplotlib inline

df_cts["Adj Close"].plot(grid = True)
```

Out[10]:

<AxesSubplot:>

In [11]:

```
s="INFY"
infosys, cts = (pdr.get_data_yahoo(s, start="2018-01-01").reset_index() for s in ["INFY"
, "CTS"])
```

```
stocks = pd.DataFrame({"INFY": df_info["Adj Close"],
                       "CTS": df_cts["Adj Close"]})

stocks.head()
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
```

Out[11]:

	INFY	CTS
0	**7.429263**	**26.042004**
1	**7.365414**	**25.796324**
2	**7.319807**	**26.287683**
3	**7.456626**	**26.385950**
4	**7.515914**	**26.238546**

In [12]:

```
stocks.plot(grid = True)
```

Out[12]:

```
<AxesSubplot:>
```

LAB -12

Question1. Perform CRUD operations on Student Table as outlined in the reference

https://medium.com/analytics-vidhya/programming-with-databases-in-python-using-sqlite-4cecbef51ab9

In [2]:

```
import sqlite3
conn = sqlite3.connect('my_database.sqlite')
cursor = conn.cursor()
print("Opened database successfully")
```

```
Opened database successfully
```

Creating a Table

In [4]:

```
cursor.execute('''CREATE TABLE SCHOOL
         (ID INT PRIMARY KEY     NOT NULL,
         NAME           TEXT    NOT NULL,
         AGE            INT     NOT NULL,
         ADDRESS        CHAR(50),
         MARKS          INT);''')
cursor.close()
```

In [7]:

```
import sqlite3
conn = sqlite3.connect('my_database.sqlite')
cursor = conn.cursor()

cursor.execute("INSERT INTO SCHOOL (ID,NAME,AGE,ADDRESS,MARKS) \
      VALUES (1, 'Rohan', 14, 'Delhi', 200)");
cursor.execute("INSERT INTO SCHOOL (ID,NAME,AGE,ADDRESS,MARKS) \
      VALUES (2, 'Allen', 14, 'Bangalore', 150 )");
cursor.execute("INSERT INTO SCHOOL (ID,NAME,AGE,ADDRESS,MARKS) \
      VALUES (3, 'Martha', 15, 'Hyderabad', 200 )");
cursor.execute("INSERT INTO SCHOOL (ID,NAME,AGE,ADDRESS,MARKS) \
      VALUES (4, 'Palak', 15, 'Kolkata', 650)");conn.commit()
conn.close()
```

SELECTING records from the TABLE

In [9]:

```
import sqlite3
conn = sqlite3.connect('my_database.sqlite')
cursor = conn.cursor()
for row in cursor.execute("SELECT id, name, marks from SCHOOL"):
    print("ID = ", row[0])
    print("NAME = ", row[1])
    print("MARKS = ", row[2], "\n")

conn.commit()
conn.close()
```

```
ID =  1
NAME =  Rohan
MARKS =  200
```

```
MARKS =  200

ID =  2
NAME =  Allen
MARKS =  150

ID =  3
NAME =  Martha
MARKS =  200

ID =  4
NAME =  Palak
MARKS =  650
```

UPDATING Records in the TABLE

In [10]:

```
import sqlite3
conn = sqlite3.connect('my_database.sqlite')
cursor = conn.cursor()
conn.execute("UPDATE SCHOOL set MARKS = 250 where ID = 3")
conn.commit()

for row in cursor.execute("SELECT id, name, address, marks from SCHOOL"):
   print("ID = ", row[0])
   print("NAME = ", row[1])
   print("MARKS = ", row[2], "\n")

conn.commit()
conn.close()
```

```
ID =  1
NAME =  Rohan
MARKS =  Delhi

ID =  2
NAME =  Allen
MARKS =  Bangalore

ID =  3
NAME =  Martha
MARKS =  Hyderabad

ID =  4
NAME =  Palak
MARKS =  Kolkata
```

DELETE Operation

In [14]:

```
import sqlite3
conn = sqlite3.connect('my_database.sqlite')
cursor = conn.cursor()
conn.execute("DELETE from  SCHOOL where ID = 2")
conn.commit()

for row in cursor.execute("SELECT id, name, address, marks from SCHOOL"):
   print("ID = ", row[0])
   print("NAME = ", row[1])
   print("ADDRESS = ", row[2])
   print("MARKS = ", row[3], "\n")

conn.commit()
conn.close()
```

```
ID =  1
NAME =  Rohan
ADDRESS =  Delhi
MARKS =  200

ID =  3
NAME =  Martha
ADDRESS =  Hyderabad
MARKS =  250

ID =  4
NAME =  Palak
ADDRESS =  Kolkata
MARKS =  650
```

Question2. Open the table MyRestaurants.db that you have created for NoSQL course

In [52]:

```
import cx_Oracle
conn = cx_Oracle.connect( "scott/scott")
sql = "SELECT * FROM MY_RES "
cursor = conn.cursor()
cursor.execute(sql)
for row in cursor.execute("SELECT * from MY_RES"):
    print("NAME = ", row[0])
    print("FOODTYPE=",row[1])
    print("DISTANCE=" ,row[2])
    print("LASTVISIT=",row[3])
    print("ILIKE=",row[4])
    print("\n")

conn.commit()
conn.close()
```

```
NAME =  dosacorner
FOODTYPE= nonveg
DISTANCE= 10
LASTVISIT= 05-feb-2020
ILIKE= 1


NAME =  apple_leaf
FOODTYPE= nonveg
DISTANCE= 15
LASTVISIT= 01-jan-2020
ILIKE= 1


NAME =  sowmyas
FOODTYPE= veg
DISTANCE= 18
LASTVISIT= 20-mar-2020
ILIKE= 1


NAME =  thinnappa
FOODTYPE= nonveg
DISTANCE= 25
LASTVISIT= 20-nov-2019
ILIKE= 0


NAME =  sribhavan
FOODTYPE= veg
DISTANCE= 18
LASTVISIT= 20-dec-2019
ILIKE= 0
```

```
NAME =  chinaworld
FOODTYPE= chinese
DISTANCE= 14
LASTVISIT= 05-mar-2020
ILIKE= 1


NAME =  littlechina
FOODTYPE= chinese
DISTANCE= 30
LASTVISIT= 10-mar-2020
ILIKE= 0


NAME =  munivilas
FOODTYPE= nonveg
DISTANCE= 20
LASTVISIT= 05-dec-2019
ILIKE= None
```

Question3. Write a SQL query that returns all restaurants in your table MyRestaurants.db.

In [53]:

```
import cx_Oracle
conn = cx_Oracle.connect( "scott/scott")
sql = "SELECT * FROM MY_RES "
cursor = conn.cursor()
cursor.execute(sql)
for row in cursor.execute("SELECT NAME from MY_RES"):
    print("NAME = ", row[0])
conn.commit()
conn.close()
```

```
NAME =  dosacorner
NAME =  apple_leaf
NAME =  sowmyas
NAME =  thinnappa
NAME =  sribhavan
NAME =  chinaworld
NAME =  littlechina
NAME =  munivilas
```

Question4. Write a SQL query that returns the names of restaurants in descending order that makes Chinese foods.

In [54]:

```
import cx_Oracle
conn = cx_Oracle.connect( "scott/scott")
sql = "SELECT * FROM MY_RES "
cursor = conn.cursor()
cursor.execute(sql)
for row in cursor.execute("SELECT NAME, FOODTYPE from MY_RES WHERE FOODTYPE = 'chinese'
GROUP BY NAME, FOODTYPE ORDER BY NAME, FOODTYPE DESC"):
    print("NAME = ", row[0])
    print("FOODTYPE= ", row[1])
    print("\n")
conn.commit()
conn.close()
```

```
NAME =  chinaworld
FOODTYPE=  chinese
```

```
NAME =  littlechina
FOODTYPE=  chinese
```

In [ ]:

```
```

LAB - 13

2D and 3D Data Visualization using Matplotlib and Seaborn

In this lab, you will draw simple 2-dimensional and 3-dimensional charts from matplotlib and seaborn packages.

Question1. Plot all 2D and 3D Plots using Matplotlib and Seaborn.

*** Plot 2D line, bar, histogram and box plot using Matplotlib**

*** Histogram and box plot using Seaborn**

Reference1: https://acadgild.com/blog/data-visualization-using-matplotlib-and-seaborn

Reference2: https://www.kaggle.com/janani90/data-viz-exercise-matplotlib-and-seaborn

In [7]:

```
import seaborn as sns #dataviz modules
import pandas as pd #data_processing modules
import matplotlib.pyplot as plt #data_viz
%matplotlib inline
```

In [10]:

```
data_heart = pd.read_csv('heart.csv')
data_heart.head()
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [11]:

```
data_heart.dtypes
```

Out[11]:

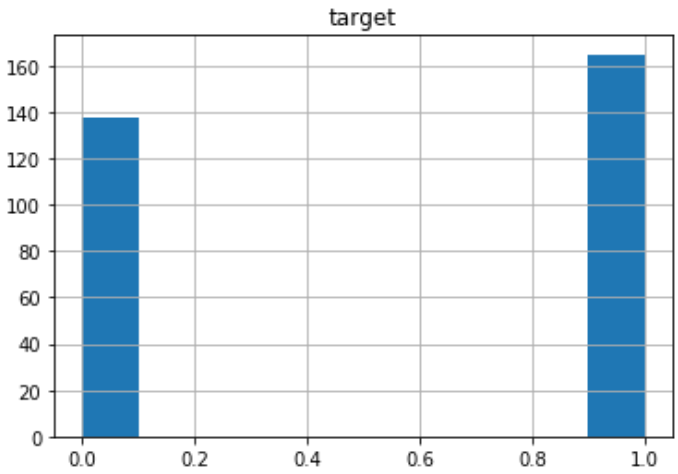
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [3]:

```
data_heart.hist('target')
```

Out[3]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f013e7cb048>]],
      dtype=object)
```

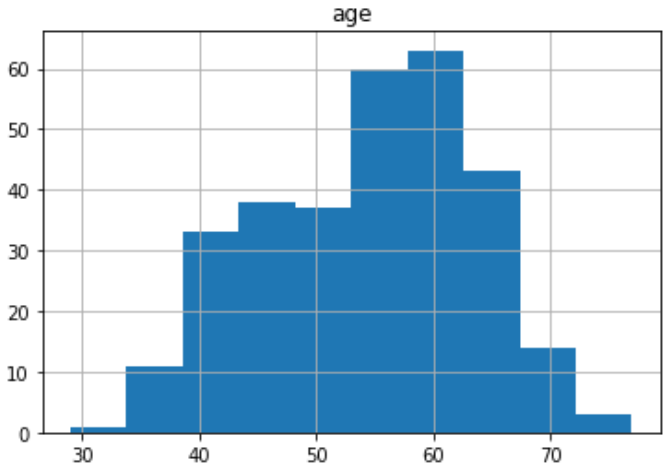


In [4]:

```
data_heart.hist('age') # histogram for one quantitative variable
```

Out[4]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f013e77eeb8>]],
      dtype=object)
```



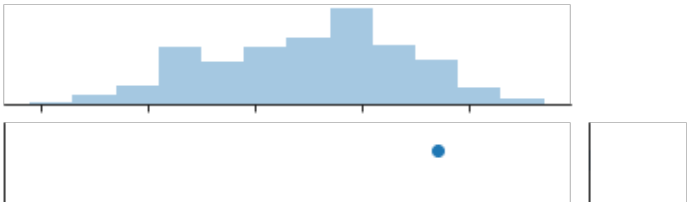
In [5]:

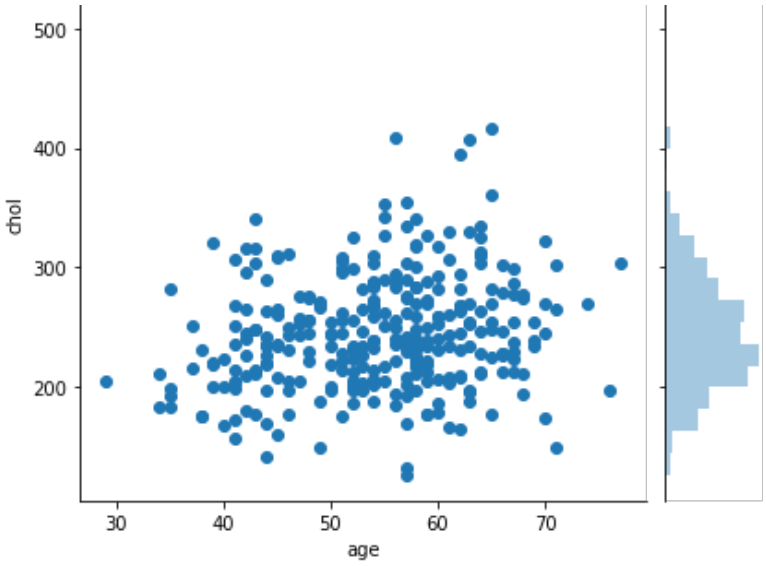
```
sns.jointplot(x='age',y='chol',data=data_heart) # scatter plot for 2 quantitative variabl
es
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a
non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` ins
tead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.arr
ay(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[5]:

```
<seaborn.axisgrid.JointGrid at 0x7f013e775e80>
```



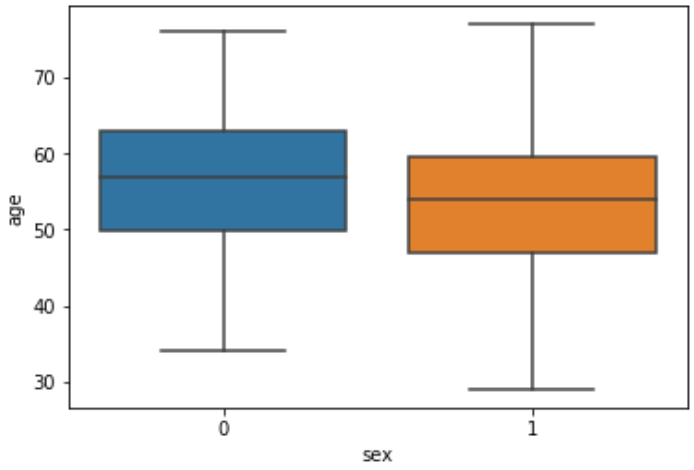


In [6]:

```
sns.boxplot(x='sex',y='age',data=data_heart)
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f013ad27eb8>
```



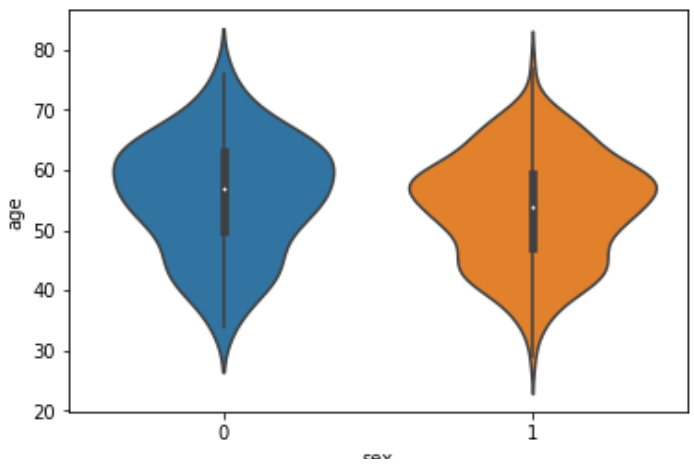
In [7]:

```
sns.violinplot(x='sex',y='age',data=data_heart)
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a
non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` ins
tead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.arr
ay(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f013acb0208>
```



In [8]:

```
data_heart.corr()
```

Out[8]:

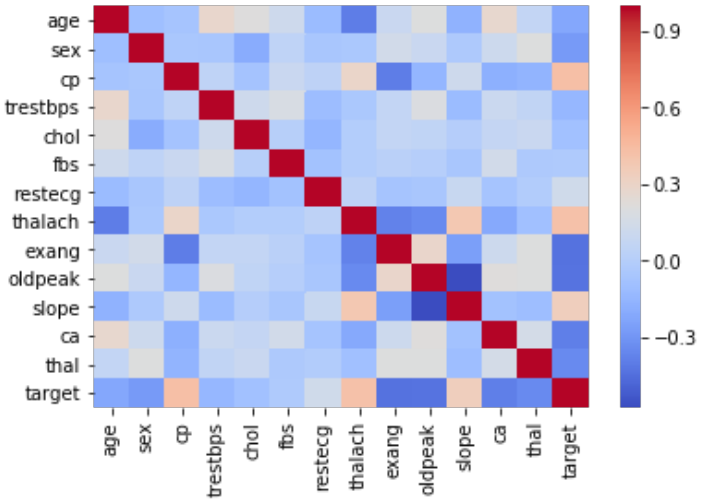
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	0.181
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	0.072
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	0.213
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	0.080
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	0.391

In [9]:

```
sns.heatmap(data_heart.corr(),cmap='coolwarm')
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f013845ee48>
```



```r
install.packages("gapminder")
install.packages("ggplot2")
install.packages("gganimate")
library(gapminder)
library(ggplot2)# data visualisation
library(gganimate)# animation
gapminder
dim(gapminder)# dimensions
str(gapminder)
#histogram/density plot
ggplot(gapminder,aes(gdpPercap))+geom_histogram()
#dataset
#aesthetic-aes()
#geometry-geom()
#facet-subplots

#scatterplot
ggplot(gapminder,aes(gdpPercap,lifeExp,size=pop))+
  geom_point(aes(color=continent),show.legend = FALSE)+
  scale_size(range = c(2,12))+
  scale_x_log10() +
  facet_wrap(continent~.)+
  labs(title = 'Year: {frame_time}', x = 'GDP per capita', y = 'life expectancy') +
  transition_time(year) +
  ease_aes('linear')
#anim <- animate(p)
#magick::image_write(anim, path="myanimation.gif")


```