1. Write a program to prompt the user for hours and rate per hour using input to compute gross pay. Pay the hourly rate for the hours up to 40 and 1.5 times the hourly rate for all hours worked above 40 hours. Use 45 hours and a rate of 10.50 per hour to test the program (the pay should be 498.75). You should use input to read a string and float() to convert the string to a number.
2. Write a program to prompt for a score between 0.0 and 1.0. If the score is out of range, print an error. If the score is between 0.0 and 1.0, print a grade using the following table:

Score Grade

>= 0.9 A

>= 0.8 B

>= 0.7 C

>= 0.6 D

< 0.6 F

If the user enters a value out of range, print a suitable error message and exit. For the test, enter a score of 0.85.

1. Write a program to prompt the user for hours and rate per hour using input to compute gross pay. Pay should be the normal rate for hours up to 40 and time-and-a-half for the hourly rate for all hours worked above 40 hours. Put the logic to do the computation of pay in a function called computepay() and use the function to do the computation. The function should return a value. Use 45 hours and a rate of 10.50 per hour to test the program (the pay should be 498.75). You should use input to read a string and float() to convert the string to a number.
2. Finding Numbers in a Haystack

In this assignment you will read through and parse a file with text and numbers. You will extract all the numbers in the file and compute the sum of the numbers.

Data Files

We provide two files for this assignment. One is a sample file where we give you the sum for your testing and the other is the actual data you need to process for the assignment.

Sample data: <http://py4e-data.dr-chuck.net/regex_sum_42.txt> (There are 90 values with a sum=445833)

Actual data: <http://py4e-data.dr-chuck.net/regex_sum_57123.txt> (There are 49 values and the sum ends with 560)

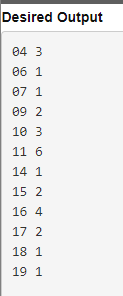
These links open in a new window. Make sure to save the file into the same folder as you will be writing your Python program. Note: Each student will have a distinct data file for the assignment - so only use your own data file for analysis.

1. Write a program to read through the mbox-short.txt and figure out the distribution by hour of the day for each of the messages. You can pull the hour out from the 'From ' line by finding the time and then splitting the string a second time using a colon.

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

Once you have accumulated the counts for each hour, print out the counts, sorted by hour as shown below.

Mbox-short.txt file link: <https://www.py4e.com/code3/mbox-short.txt>



1. Write a program that repeatedly prompts a user for integer numbers until the user enters 'done'. Once 'done' is entered, print out the largest and smallest of the numbers. If the user enters anything other than a valid number catch it with a try/except and put out an appropriate message and ignore the number. Enter 7, 2, bob, 10, and 4 and match the output below.
2. 6.5 Write code using find() and string slicing (see section 6.10) to extract the number at the end of the line below. Convert the extracted value to a floating point number and print it out.
3. Write a program that prompts for a file name, then opens that file and reads through the file, looking for lines of the form:

X-DSPAM-Confidence: 0.8475

Count these lines and extract the floating point values from each of the lines and compute the average of those values and produce an output as shown below. Do not use the sum() function or a variable named sum in your solution.

You can download the sample data at <http://www.py4e.com/code3/mbox-short.txt> when you are testing below enter mbox-short.txt as the file name.

1. Open the file romeo.txt and read it line by line. For each line, split the line into a list of words using the split() method. The program should build a list of words. For each word on each line check to see if the word is already in the list and if not append it to the list. When the program completes, sort and print the resulting words in alphabetical order.

You can download the sample data at <http://www.py4e.com/code3/romeo.txt>

1. Open the file mbox-short.txt and read it line by line. When you find a line that starts with 'From ' like the following line:

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

You will parse the From line using split() and print out the second word in the line (i.e. the entire address of the person who sent the message). Then print out a count at the end.

Hint: make sure not to include the lines that start with 'From:'.

You can download the sample data at <http://www.py4e.com/code3/mbox-short.txt>

1. Write a program to read through the mbox-short.txt and figure out who has the sent the greatest number of mail messages. The program looks for 'From ' lines and takes the second word of those lines as the person who sent the mail. The program creates a Python dictionary that maps the sender's mail address to a count of the number of times they appear in the file. After the dictionary is produced, the program reads through the dictionary using a maximum loop to find the most prolific committer.