1) Write a python program to prompt a user to enter a low and a high value.  Your program is then to output all the **even** values between and including the low value and the high value, a bar and then the final sum.  If the low value is greater than the high value, or there are no even values in the specified interval, the program should output “There are no even values in this interval’. So for instance, if the user entered low as 2 and high as 9, your program should output

2

4

6

8

-----

20

2)     A prime number is a number that  can only be evenly  divided by  1 and itself (like 17).  Prompt the user to enter a number and your program should determine if the number is a prime number or not.  This is a trial and error problem, but your methodology should be efficient.  For instance, once it has been determined that a number is not prime, your loop should terminate.  Also, consider what values do not need be checked and  design your algorithm so that you do not check them.  For instance ask yourself, “Might 5 evenly  divide 17?”  Well, maybe, so it should be checked, because if it evenly divides 17 then 17 would not be prime.  On the other hand, what is the answer to “Might 14 evenly divide 17?” and how should that effect the efficient design of your algorithm.

3)  Prompt the user to enter an integer between 1 and 10.  If they enter a number out of that range, print an error statement and prompt them again to enter a number in the correct range.  However, they should get no more than 5 tries.  If they have not entered a correct number after 5 tries, a final error message should be output and the program should terminate. If they do enter a correct number, simply output that number.

4)     Prompt the user to enter an integer.  Your program is to print out every number that evenly divides that integer.

5)  Write a program to prompt the user to enter the number of questions on an exam, then prompt the user to enter the scores for that exam print out the total.  Update the program so that it asks the user to input the number of students in the class, and the number of questions on the exam, and then prompts the user to enter the exam scores and gives a total for each student in the class.  As an extra, also calculate and output the average test scores of all the students.

Here are some practice Python questions for Tuesday's quiz, which focus on strings, lists, functions and file i/o.

**You should know common string and list methods, functions and operators by heart.**

6) Assume the existence of a list called *deck*, containing strings representing cards where each card is represented by a string consisting of a single character representing the suit (S,H,D,C) followed by a value 2-10,J,Q,K,A.

Write a code segment that will “deal” the “cards” from *deck* alternately into two other lists, *player1* and *player 2*.  When the code segment is complete, *deck* should be empty.  The first card that was originally on the deck will be at the last item of *player1*’s list, the second card that was originally on the deck will be at the last item  of *player2*’s list etc., so that the player1 and player2 represent their cards as if they were dealt to them in a pile.

7) Assume the existence of a list of integers called *num\_list*.  Write a code segment that will remove every other item from the list, so if *num\_list* originally contained [3,6,9,4,1], it will then contain [3,9,1].

8) You are to assume the existence a variable *numlist* which is a list of integers and assume the existence of two variables, x and y, that contain integers. Write a code segment that will change the value in the location in *numlist* indexed by *x* to the value in location *y*.  For example, if x is 4 and y is 8, the location indexed by 4 in *numlist*should be assigned the value in location 8.  However, before making that change, it should be insured that the indices specified by the x and y are not higher than the highest index in *numlist*.  If it is, the error message “index too large” should be output and no change should be made to *numlist*.

9) Assume the existence of a list of integers *numlist*. Write a code segment that will remove the three highest numbers from the list without changing the order of the elements in the list. You may assume that there are at least 3 elements in the list.

10) Assume the existence of a list *friends* where each element contains a string. Write a code segment that will prompt the user to enter a name. The code should then remove all occurrences of that string from the list (so more than one element may contain the string). If there had been elements that contained the string, the number of elements that were removed should be output.