Assignment 4: Functions

Due Date: Nov 12th Sunday, 11.59pm

Python Library Functions allowed:

len(),

print(),

list(),

str(),

input(),

int(),

type(),

chr(),

ord()

float().

randint()

random()

Libraries allowed: random, math. All functions in these packages are allowed as well.

List functions allowed: append(), replace(), remove(), index() and len()

List indexing: L[0], L[1],... to refer to the list values.

Grading Emphasis: Input validation, error checking and well commented logically correct programs.

Submissions format accepted:

1. Type in or copy the python scripts into BlackBoard.

Or

2. Upload the individual python files by naming them with the question numbers.

3. Please DO NOT copy the questions into your submissions. Just make sure to number your answers correctly.

Questions

1. A prime number is a number that is only evenly divisible by itself and 1. For example, the

number 5 is prime because it can only be evenly divided by 1 and 5. The number 6, how-

ever, is not prime because it can be divided evenly by 1, 2, 3, and 6. Write a function to take a number N as input from the user and output whether the number is prime. The following functions should be in the submitted Python script:

is\_prime(N): a function that N as argument and outputs whether the number is prime or not.

main(): the function where the program execution should begin and end.

2. Write a program that generates N random numbers between 1 and 1000. The program should print out a number only if it is a prime number. Once N numbers are printed, the program should stop. Use the is\_prime() function in Problem 1 for checking the prime number validity. The value of N should be taken as input from the user. The following functions should be in the submitted Python script:

is\_prime(): function to check whether a number N is prime or not

random\_gen(): function that generates one random number for processing

main(): the function where the program execution should begin and end.

The number of function arguments and parameters are left for your own design choice.

3. Write a program that generates a random number in the range of 1 through 1000, and asks

the user to guess what the number is. If the user’s guess is higher than the random number,

the program should display “Too high, try again.” If the user’s guess is lower than the

random number, the program should display “Too low, try again.” If the user guesses the

number, the application should congratulate the user and generate a new random number

so the game can start over. If the user doesn't want to continue the game the user should type an appropriate stop symbol to terminate the program (such stop conditions were discussed in the class). For each game iteration, count the number of guesses made by the user and display them. The program should contain the following functions:

check\_guess(): this function checks if the user's guess is lower, higher or equal to the random number.

random\_gen(): function that generates one random number for one game iteration

main(): the function where the program execution should begin and end.

The number of function arguments and parameters are left for your own design choice.

4. Write a program that asks user to input negative and positive integers between -10 and 10. The program will add up the numbers and terminate when the sum of the values becomes zero. The program should contain the following functions:

add() : add up the values

main(): the function where the program execution should begin and end.

A global variable subsetsum that keeps track of the sum

The number of function arguments and parameters are left for your own design choice.

5. Repeat the task above by using the randint function to generate the random numbers between -25 and 25. The program should contain the following functions.

add() : add up the values

main(): the function where the program execution should begin and end.

randomgen(): a function that returns a number between -25 and 25

A global variable subsetsum that keeps track of the sum

The program should output the count of the number of values generated before exiting saying that the subset sum was zero.