CS217: Computer Programming Language: Iteration 1

Instructor: Jakramate Bootkrajang

Simulation

- To simulate natural phenomenon
- One of the useful functions is random number generating function “random()” from random package

```
[3]  from random import random

     value = random()  # return number from 0 to 1

     print(value)
```

```
0.6533677130501213
```

Exercise 1

- Write a function that simulate a dice roll. The function should return number 1,2,3,4,5 or 6
- Hint: divide [0,1] returns from random() into 6 equally spaced range.

Exercise 2

- Use the function you've wrote to, write a program to calculate the probability of rolling two dices and get the sum greater than 10

Exercise 61: Average

(*26 Lines*)

In this exercise you will create a program that computes the average of a collection of values entered by the user. The user will enter 0 as a sentinel value to indicate that no further values will be provided. Your program should display an appropriate error message if the first value entered by the user is 0.

> Hint: Because the 0 marks the end of the input it should **not** be included in the average.

Exercise 67: Admission Price

(*Solved—38 Lines*)

A particular zoo determines the price of admission based on the age of the guest. Guests 2 years of age and less are admitted without charge. Children between 3 and 12 years of age cost $14.00. Seniors aged 65 years and over cost $18.00. Admission for all other guests is $23.00.

Create a program that begins by reading the ages of all of the guests in a group from the user, with one age entered on each line. The user will enter a blank line to indicate that there are no more guests in the group. Then your program should display the admission cost for the group with an appropriate message. The cost should be displayed using two decimal places.

Exercise 71: Square Root

(14 Lines)

Write a program that implements Newton's method to compute and display the square root of a number entered by the user. The algorithm for Newton's method follows:

Read x from the user
Initialize *guess* to $x/2$
While *guess* is not good enough **do**
    Update *guess* to be the average of *guess* and $x/guess$

When this algorithm completes, *guess* contains an approximation of the square root. The quality of the approximation depends on how you define "good enough". In the author's solution, *guess* was considered good enough when the absolute value of the difference between $guess * guess$ and x was less than or equal to 10^{-12}.