

Python Programming for All Lab 6

INSTRUCTOR: Dr. Albert Kahira albert@jengaschool.com

LOOPS

- 1. We will compare the control of the start, end, and step values in a counter loop, both in for and while loops
 - **a.** Write a function that finds the sum of numbers from 1 to 100 (both included) using a for loop. Write another function that does the same operation using a while loop.
 - **b.** Modify your functions such that summation starts from 5 instead of 1. Then further generalize such that the start value is entered by the user.
 - **c.** Modify both functions such that the end value is not 100, but it is entered by the user. You can use either < or <= in the while condition. Consider both cases, how would the boundary value change for both cases?
 - **d.** Modify the functions such that numbers increment by 2.
- 2. If you continually divide a number by 2, it will equal to zero after you do the division an infinite number of times: The sequence 1, ½, ¼, 1/8, 1/16, ... 1/2n approaches 0 as n approaches infinity. However, when you use a float, because of the rounding errors, the result becomes zero after a number of steps (before infinity). Let's see this using a while loop.
 - **a.** In the beginning of the program, initialize a float variable to 1 (e.g. x=1.0). Then, in a while loop, continually divide x by 2 (x=x/2). When the loop

- finishes, the program should print an appropriate message. Do you think this loop will stop or will it run forever?
- **b.** Add an appropriate print statement in the loop body so that you can see how the value of x changes.
- **c.** Add a counter to your program and find in how many steps x reaches the value 0.
- **d.** Comment on what would be the difference, in general, between using x>0 or x!=0 as the while condition, when x is a float. Note that floats are not always exact.
- 3. You may have to be careful when predicting the value of your counter (or any other variable that is modified in the loop body) when you exit a while loop.
 - **a.** In the following program, what is the value of **i** after the loop finishes (final value)? What would be the final value of **i** for the cases where you decrement **i** by 2, 4, 5, and 10 (instead of 3)?

```
def finalval():
    i = 25
    while i>12:
        print(i)
        i = i-3
    print("Final Value: ", i)
finalval()
```

b. In the following program, what is the value of x after the loop finishes (final value)? What would be the final value of x for the cases where the initial value of x is 6, 5, and 8?

```
def finalval2():
    x = 7
    while x>= 5 and x<=8:
        print(x)
        if x%2 == 0:
            x = x+1
        else:
            x = x-2
        print("Final Value: ", x)</pre>
```

- 4. A user wants to purchase a list of items as much as the budget allows. Let's say the budget is 100TL. Write a program that reads item prices from the user one by one; prints the total up to that point after reading each price; and prints a message and stops when the total exceeds 100TL.
- 5. We can use a while loop to make sure that the user inputs a value that satisfies certain conditions. We made a similar check before using if statements. But in order to be able to ask for a new input, you need a loop. Checking that an input satisfies certain conditions (e.g. a number is positive, the password is long enough, the date is valid, etc.) is called 'input validation'.
 - **a.** In the following program, input validation is used to make sure the number for the square root is non-negative. A while loop is used to wait until a valid input is entered.

```
def valid_squareroot():
    x = int(input("Enter the number for SquareRoot:"))
    while x<0:
        print("Number cannot be negative...")
        x = int(input("Enter the number for SquareRoot:"))
        print("The squareroot of", x, "is", x**0.5)
valid_squareroot()</pre>
```

b. Based on the above example write a function that asks the user for a password, and repeats this until the user enters a password that you determine.