Lecture 5
Repetition

Objectives

- After completing the lesson, the student will be able to:
  - Explain the syntax of for loop.
  - Write programs using for statements and run the program.
  - Explain the syntax of while loop.
  - Write programs using while statement and run the program.
  - Describe the output of programs with repetition.

Repetition Control Structures

- Statements that allows to execute specific block of code a number of times.

- There are two types:
  - For loop - used to iterate over a sequence of elements.

  - While loop - an indefinite loop which keeps iterating until certain conditions are met.

For Loop

- General Syntax:

```
for var in sequence:
   body
else:
   post-termination
```

- The `else` clause is optional.

For Loop

- Variable - takes on the consecutive values from the list until the sequence is exhausted.
- Sequence - can be a literal sequence (list or tuple etc…).
- Body - Python code executed on each iteration.
- post-termination - Python code executed after the sequence has been exhausted.

For Loop - Iterating Over a Sequence

```
numbers = [2,4,6,8,10,12,14,16,18,20]
for num in numbers:
    print(num)
else:
    print("List of even numbers")
```

For Loop - Iterating Over a Sequence

```
List = 
[2,"Two",4,"Four",6,"Six",8,"Eight",10,"Ten"]
for value in List:
    print(value)
else:
    print("List with different values")
```

For Loop - Iterating Over a Sequence

```
fruits = ["apple", "pear", "mango", "kiwi"]
for f in fruits:
    print(f)
else:
    print("\n Done with the fruits")
```

The range() Function

- `range([start], stop, [step])`
- It creates lists of integers in an arithmetic progression. It is primarily used in for loops. The start parameter is optional and defaults to 0 when not specified. Similarly, step defaults to 1.

- Example:
  - `range(10)`, returns [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  - `range(10,15)`, returns [10, 11, 12, 13, 14]
  - `range(10,20,2)`, returns [10, 12, 14, 16, 18]

Example – Calculate the average of numbers

- Here is the design to calculate average of some numbers.

- Input the count of the numbers, n
- Initialize 0 to sum
- Loop n times
  - Input a number, x
  - Add x to sum
- Output average as sum / n

Example – Calculate the average of numbers

- The above design can be translated to a program.

```
n = int(input("How many numbers do you have? "))
sum = 0.0
for i in range(n):
    x = int(input("Enter a number: "))
    sum = sum + x
print("\nThe average of the numbers is", sum / n)
```

Some Points to Remember

- Proper indentation should be used to get the correct output of the program.
- `\n` is used in the last statement to print a blank line before printing the output.
- Comma (,) is used in the last statement to separate the message and the value of the variable.

Output of the Program

- How many numbers do you have? 5
- Enter a number >> 5
- Enter a number >> 10
- Enter a number >> 15
- Enter a number >> 20
- Enter a number >> 25

- The average of the numbers is 15.0

Counting with a for Loop

```
for x in range(1,6):
    print(x)
else:
    print("We counted to 5!")
```

- Output
  - 1
  - 2
  - 3
  - 4
  - 5
  - We counted to 5!

While Loop

- General Syntax:

```
while  condition:
   body
else:
   post-termination
```

- The else clause is optional.
- Condition is a Boolean expression.
- Body - Python code executed on each iteration where variable can be used.
- post-termination - Python code executed after the sequence has been exhausted.

While Loop

- The body of the loop executes repeatedly as long as the condition remains true.
- When the condition is false, the loop terminates.
- In pre-test loops, condition is always tested at the beginning of the loop.
- If the loop condition is initially false, the loop body will not execute at all.

Example – List Numbers in Reverse Order

```
x = 5
while x > 0:
   print(x)
   x = x - 1
else:
   print("We counted backwards from
   5!")
```

Output – List Numbers in Reverse Order

- 5
- 4
- 3
- 2
- 1
- We counted backwards from 5!

Example – List Even Numbers Below 10

```
x = 2
while x < 10:
  print(x)
  x = x + 2
else:
  print(" List of even numbers below 10")
```

Output– List Even Numbers Below 10

- 2
- 4
- 6
- 8
- List of even numbers below 10

While loop - break

- The `break` is a one-line statement that means "exit the current loop."
- An alternate way to make a loop exit and stop executing is with the break statement.
- First, create a `while` with a condition that is always true.
- Using an `if` statement, define the stopping condition. Inside the `if`, write `break`, meaning "exit the loop.”

While loop – break - Example

```
x = 10
while x > 0:
    print(x)
    x = x - 1
    if x == 5:
        break
else:
    print("We counted down from 10!")
```

Some Points to Remember

- Remember to indent the statements after the condition and else statement.
- While loop creates infinite loops.
- Click Ctrl + C, to break out the loop.
- Click Ctrl + Alt + Delete, if the loop is really tight.

Example: Infinity Loop

- Before correcting the error

```
n = 0
while n<=10:
    print(" This is an infinity loop")
```

- After correcting the error

```
n = 0
while n < 10:
    print(" This is an infinity loop")
    n = n + 1
```