CSE 453

Loops
(Repetition Structures)

Conditional Loop

- while Loop
- Format

  while *condition*:

  statements

  - When the condition is true, the statements are executed
    - Conditions are the same as for the selection statements
  - All statements must be indented
    - Forms the block structuring
  - Note case sensitivity!
- Interactive Python Example
  - The following example keeps a computes a running sum of user-entered numbers. A sentinel value (0) used to end the loop

```
>>> num=int(input('Enter a number (zero to exit):'))
>>> sum=num
>>> while num != 0:
...     num=int(input('Enter a number (zero to exit):'))
...     sum=sum+num
...
Enter a number(zero to exit): 4
Enter a number(zero to exit): 9
Enter a number(zero to exit): 8
>>>
```

Unconditional Loop

- for Loop
- Two Variants
  - List
  - Range
- For Every Value in a List
  - Format

    for *variable* in [*value_list*]:

    statements
  - The statements are executed for each *value* in the comma separated *value_list*
    - During an iteration, the *variable* takes on the *value* associated with that iteration
  - All statements must be indented
    - Forms the block structuring
  - Note case sensitivity!

- Interactive Python Example
  - The following example assigns the variable *department* to each value in the list

```
>>> for department in ['CSE', 'EE', 'MAE']:
...     print(department)
...
CSE
EE
MAE
>>>
```

- For Every Value in a Range
  - Format

    for *variable* in range(*argument_list*):

        statements
  - The statements are executed for each *value* in a range, as specified using the built-in range function
    - During an iteration, the *variable* takes on the *value* associated with that iteration
  - All statements must be indented
    - Forms the block structuring
  - Note case sensitivity!
  - Built-in Function: range(*argument_list*)
    - One Argument Variant
      - ✓ range(*ending_value*)
        - The *statements* in the body of the loop are executed for every value in the range up to but NOT including the *ending_value*
        - The variable starts at 0 and is incremented by 1 for every iteration of the loop
    - Two Argument Variant
      - ✓ range(*starting_value*, *ending_value*)
        - The *statements* in the body of the loop are executed for every value in the range up to but NOT including the *ending_value*
        - The variable starts at the *starting_value* and ends at the *ending_value*
    - Three Argument Variant
      - ✓ range(*starting_value*, *ending_value* , *step_value*)
      - ✓ The variable starts at the *starting_value* and ends at the *ending_value*
        - The *statements* in the body of the loop are executed for every value in the range up to but NOT including the *ending_value*
        - The variable is incremented by the *step_value* for every iteration of the loop

- Example Using the range() Function

```
>>> for counter in range(40,100,10):
...     print(counter)
...
40
50
60
70
80
90
>>>
```

Nested Loops

- The key to nesting a loop within a loop is the indenting!
- Example

```
number=1
while number > 0:
        number = 0
        number=int(input("Enter a number: "))
        factorial = 1
        for counter in range (1,(number+1)):
             factorial = factorial * counter
        print("The factorial of ",number," is ",factorial)
```

References

- Tony Gaddis, *Starting Out With Python*, Second Edition, Pearson Education, Inc. (Addison Wesley), 2012