The *for* Loop

Section 4

Chapter 3

Quiz 7

The *for* Loop

- A *for* loop is used to iterate through a sequence of values

```
for var in sequence:
    indented block of statements
```

- The sequence in a for loop can be
  - An arithmetic progression of numbers
  - A string
  - A list
  - A tuple

- Physical indentations tell interpreter where block starts and stops

Looping through Arithmetic Progression

The range Function

- The range function is used to generate an arithmetic progression
- range(m, n) generates a sequence m, m+1, …, n-1

`range(3, 10)` generates the sequence $3, 4, 5, 6, 7, 8, 9$.

`range(0, 4)` generates the sequence $0, 1, 2, 3$.

`range(-4, 2)` generates the sequence $-4, -3, -2, -1, 0, 1$.

```
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> list(range(1, 5))
[1, 2, 3, 4]
```

Step Values for the range Function

- range(m, n, s) generates m, m+s, m+2s, …, m+rs, where r is the largest whole number so that m+rs < n.

- If a negative step value is used when the initial value is greater than terminating value,

```
>>> list(range(0, 5, 2))
[0, 2, 4]
>>> list(range(1, 5, 2))
[1, 3]
```

```
>>> list(range(5, 0, -1))
[5, 4, 3, 2, 1]
>>> list(range(5, 0, -2))
[5, 3, 1]
```

Example 1: Display numbers 1-5

- Displays numbers from 1 to 5.
  - It would be tedious to use print(1), print(2), print(3), …
  - Use a repetition structure instead.
  - We used a *while* loop last time. We can also use a *for* loop.

```
1
2
3
4
5
```

Example 2: Display “Hello World!” five times

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Looping over Strings

Example 3: display each character of a string

```
C
i
t
y
U
```

Example 4: reverse words

- Use a *for* loop to reverse words.

```
Give me a word: CityU
UytiC
```

```
>>> "CityU"[::-1]
'UytiC'
```

Looping over Lists and Tuples

Example 5: Month abbreviation

Abbreviate the months using the first three letters:

months = ["January", "February", "March", "April", "May", "June", \
"July", "August", "September", "October", "November", "December"]

Output:

```
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

Nested *for* Loops

- The Body of for loop can contain any type of Python statement
  - Can contain another for loop, a while loop, if statements, etc.

- The second loop must be completely contained inside the first loop
  - Must have a different loop variable

Example 6: Display the 9 by 9 multiplication table

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27

4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45

6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54

7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63

8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72

9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```

The *pass* Statement

- There are times when you want loop to cycle through a sequence and not do anything
  - The *pass* statement should be used.

- The *pass* statement is a do-nothing placeholder statement

Example 7: pass statement

```
# pass statement

for x in [1, 5, 9]:
    pass
print("Loop completed")
```

The *continue* Statement

- When **`continue`** is executed in a for loop
  - The Current iteration of the loop terminates
  - Execution returns to the loop's header

- **`continue`** is usually under an *if* statement.

Example 8: continue

```
# continue statement

for x in [1, 5, 9]:
    if x==5:
        continue
    print(x)
```

```
1
9
```

The *break* Statement

- When **`break`** is executed
  - Loop immediately terminates

- Break statements usually occur under *if* statements

Example 9: break

```
# break statement

for x in [1, 5, 9]:
    if x==5:
        break
    print(x)
```

```
1
```

Classwork 7: Present Value of Annuity Due

- *Annuity due* is an annuity with a cash flow occurring at the *beginning* of each period, as opposed to an ordinary annuity with the cash flow at the end of each period. Monthly rent is one example of annuity due - rent is due at the beginning of every month.
- Suppose the monthly rent is HKD 10,000. Assume the rent is discounted monthly with a rate of 0.25%. Write a Python program to calculate the *present value* of the total rent:
  - Let the user enter the number of years in the lease.
  - Use a *for* loop.
  - Round the answer to two decimal places. The output should resemble the following.
- Include the exit line. Upload the .py file and the output screenshot on Canvas.

```
Enter the number of years: 4
The present value of 4 year(s) of rent is $452916.41.

Press ENTER to exit.
```