1.9 Looping

IX

Repeating blocks of code by calling a function more than once, as in example 1.5, can get cumbersome when it needs to be repeated many times. A **loop** repeats a block until some stopping condition is met. One type of loop in Python is a `while` loop, which repeats a block of code while its conditional expression evaluates to `True`. For instance,

```python
n = 0           # Initialize n
while n < 5:
    print(n)
    n += 1      # Increment n (i.e., n = n + 1)
```

The loop evaluates the conditional expression `n < 5` and, if in fact `n < 5`, executes the block of code. After the block finishes, the test is repeated and potentially the block of code. This will repeat indefinitely, until the conditional expression evaluates to `False`, in which case the loop exits and execution resumes after the code block. The block will be executed 5 times, printing `0` through `4` to the console.

Another type of Python loop is a `for` loop, which has no explicit conditional expression, instead iterating through an iterable object like a list, , until it reaches the end. For example,

```python
l = ["foo", "bar", "baz"]
for s in l:
    print(f"Say {s}")
```

This prints

```
Say foo
Say bar
Say baz
```

It is common to loop through a `range` with a `for` loop, as in the following:

```python
for k in range(2, 8):
    print(k, end=" ") # Prints on the same line
```

This prints the following to the console:

```
2 3 4 5 6 7
```

Often, a loop index is required inside a `for` loop. The syntax for this requires an identifier for the index and an `enumerate` type object to be iterated through. The constructor function `enumerate()` assigns an index to each element of its iterable argument (e.g., a list). For instance,

```
names = ["Manny", "Bella", "Amadeus"]
signs = ["Libra", "Virgo", "Sagittarius"]
for i, name in enumerate(names):
    print(f"{name} is a {signs[i]}")
```

This prints the following to the console:

```
Manny is a Libra
Bella is a Virgo
Amadeus is a Sagittarius
```

Looping through a dictionary is similar, but we need the `items()` of the dictionary for the key-value pair, as follows:

```
sounds = {"dog": "woof", "cat": "meow", "fox": "ring-ding-ding"}
for k, v in sounds.items():
    print(f"The {k} says '{v}'")
```

This prints the following to the console:

```
The dog says 'woof'
The cat says 'meow'
The fox says 'ring-ding-ding'
```

1.10 Problems

6E

Problem 1.1 BA Write a program with the following requirements:

a. It defines variables for the following quantities:

$$x = 5.2 + j3.4, \quad y = -17, \text{ and } \quad z = 0.02,$$

where j is the imaginary number $\sqrt{-1}$.

b. It computes and prints the following quantities:

$$x + y, \quad xyz, \text{ and } \quad 4x^3 - 8xy + 6y^2.$$

c. It further computes and prints the following quantities:

$$|x|, \quad \overline{xy}, \text{ and } \quad \Re(x),$$

where $|\cdot|$ is the absolute value, $\overline{\ \cdot\ }$ is the complex conjugate, and $\Re(\cdot)$ is the real part.

Problem 1.2 SB Write a program with the following requirements:

a. It defines a variable for a list with the following elements:

```
4, -12, 6, -14, 8, -16
```

b. It prints the first and last elements of the list
c. Using list slicing, it prints the first three elements of the list
d. Using list slicing, it prints the last three elements of the list
e. Using list slicing, it prints every other element, starting with the first element
f. It computes and prints the length of the list (consider using the built-in function `len()`)
g. It computes and prints the sum of the list elements (consider using the built-in function `sum()`)

Problem 1.3 I2 Write a program with the following requirements:

a. It defines a variable for a list with the following elements:

```
32, 41, 58, 34, 24, 53, 46, 41
```

b. It computes and prints the mean of the list items (consider using the built-in `sum()` and `len()` functions)
c. It finds and prints the maximum and minimum values in the list (consider using the built-in `max()` and `min()` functions)