CHAPTER

FOUR

LECTURE 3 — PYTHON STRINGS

4.1 Reading

This material is drawn from Chapter 4 of *Practical Programming*, 2nd edition.

4.2 More Than Just Numbers

- Much of what we do today with computers revolves around text:
  - Web pages
  - Facebook
  - Text messages

  These require working with *strings*.
- Strings are our third type, after integers and floats.
- We've already seen the use of strings in output,

```
print("Hello world")
x = 8
y = 10
print("Value of x is", x, "value of y is", y)
```

4.3 Topics for Today

- String basics
- String operations
- Input to and output from your Python programs

4.4 Strings — Definition

- A string is a sequence of 0 or more characters delimited by single quotes or double quotes.

```
'Rensselaer'
"Albany, NY"
'4 8 15 16 23 42'
''
```

- We can print strings:

```
>>> print("Hello, world!")
Hello, world!
```

- Strings may be assigned to variables:

```
>>> s = 'Hello'
>>> t = "Good-bye"
>>> print(s)
Hello
>>> t
'Good-bye'
```

- Notice that unlike integers and floats there is now a difference between asking the Python function `print` to output the variable and asking the Python interpreter directly for the value of the variable.

4.5 Combining Single and Double Quotes in a String

- A string that starts with double quotes must end with double quotes, and therefore we can have single quotes inside.
- A string that starts with single quotes must end with single quotes and therefore we can have double quotes inside.
- To illustrate this, we will take a look at

```
>>> s = 'He said, "Hello, World!"'
>>> t = "Many single quotes here '''''''  and here ''' but correct."
```

4.6 Multi-Line Strings

- Ordinarily, strings do not extend across multiple lines, causing an error if you try.
- But, starting and ending a string `"""` or `'''` tells Python to allow the string to cross multiple lines.
    - Any character other than `'''` (or `"""`, if that is how the string started) is allowed inside the string.
- Example,

```
>>> s1 = """This
is a multi-line
string."""
>>> s1
'This\nis a multi-line\nstring.'
>>> print s1
This
is a multi-line
string.
>>>
```

- Notice the `\n` when we ask Python for the value of the string (instead of printing it). This is an *escape character*, as we will discuss next.

4.7 Escape Characters

- Inserting a \ in the middle of a string tells Python that the next character will have special meaning (if it is possible for it to have special meaning).
- Most importantly:
  - `\n` — end the current line of text and start a new one
  - `\t` — skip to the next "tab stop" in the text. This allows output in columns
  - `\'` — do not interpret the `'` as a string delimiter
  - `\"` — do not interpret the `"` as a string delimiter
  - `\\` — put a true back-slash character into the string
- We'll explore the following strings in class

```
>>> s0 = "*\t*\n**\t**\n***\t***\n"
>>> s1 = "I said, \"This is a valid string.\""
```

4.8 String Operations — Concatenation

- Concatenation: Two (or more) strings may be concatenated to form a new string, either with or without the + operator. We'll look at

```
>>> s0 = "Hello"
>>> s1 = "World"
>>> s0 + s1
>>> s0 + ' ' + s1
>>> 'Good' 'Morning' 'America!'
>>> 'Good ' 'Morning ' 'America!'
```

- Notice that

```
>>> s0 = "Hello"
>>> s1 = " World"
>>> s0 s1
```

  is a syntax error but

```
>>> "Hello" " World"
```

  is not. Can you think why?

4.9 String Operations — Replication

- You can replicate strings by multiplying them by an integer:

```
>>> s = 'Ha'
>>> print(s * 10)
HaHaHaHaHaHaHaHaHaHa
```

- What do you think multiplying a string by a negative integer or 0 does? Try it.
- Many expressions you might try to write involving strings and either ints or floats are illegal Python, including the following:

```
>>> 'Hello' * 8.1
>>> '123' + 4
```

  Think about why

4.10 Practice Problems - Part 1

We will go over these during lecture:

1. Which are valid Python strings:

```
>>> s1 = '"Hi mom", I said.  "How are you?"'
>>> s2 = '"Hi mom", I said.  '"How are you?"
>>> s3 = '"Hi mom", I said.  '"How are you?"'
>>> s4 = """'Hi mom", I said.  '"How are you?"'"""
>>> s5 = ""I want to be a lion tamer!"'
>>> s6 = "\"Is this a cheese shop?\"\n\t'Yes'\n\t\"We have all kinds!\""
```

   For those that are not valid, what needs to be fixed? For those that are, what is the output when they are passed to the `print` function?

2. What is the output?

```
>>> s = "Cats\tare\n\tgood\tsources\n\t\tof\tinternet\tmemes"
>>> s
>>> print(s)
```

3. What is the output?

```
print('\\'*4)
print('\\\n'*3)
print('Good-bye')
```

4. Which of the following are legal? For those that are, show what Python outputs when these are typed directly into the interpreter.

```
>>> 'abc' 'def'
>>> 'abc' + 'def'
>>> 'abc ' + 'def'
>>> x = 'abc'
>>> y = 'def'
>>> x+y
>>> x y
>>> s1 = 'abc'*4
>>> s1
>>> s2 = 'abc '*4
>>> print(s2)
```

4.11 String Operations — Functions

- Python provides many operations for us to use in the form of **functions**. We have already seen `print()`, but now we are going to look at other functions that operate on strings.
- You can compute the length of a string with `len()`.

```
>>> s = "Hello!"
>>> print(len(s))
```

- Here is what happens:
  1. Function `len` is provided with the value of the string associated with variable `s`
  2. `len` calculates the number of characters in the provided string using its own code, code that is *built-in* to Python.
  3. `len` *returns* the calculated value (in this case, 6) and this value is sent to the `print` function, which actually generates the output.
- We will learn more about using functions in Lectures 4 and 5.

4.12 Example String Functions

- We will look at examples of all of the following during lecture…
- You can convert an integer or float to a string with `str()`.
- You can convert a string that is in the form of an integer to an integer using `int()`
- You can convert a string that is in the form of a float to a float using, not surprisingly, `float()`

4.13 The print Function in More Detail

- We already know a bit about how to use `print()`, but we can learn about more using `help()`

```
help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

- `flush` is useful when trying to debug. If you are trying to trace your program execution using print, adding `flush=True` as your final argument will give you more accurate results. We will talk about this more later.
- For now, we will focus on the `sep` and `end` and illustrate with examples.

4.14 User Input

- Python programs can ask the user for input using the function called `input`.
- This waits for the user to type a line of input, which Python reads as a string.
- This string can be converted to an integer or a float (as long as it is properly an int/float).
- Here is a toy example

  ```
  print("Enter a number")
  x = float(input())
  print('The square of', x, 'is', x*x)
  ```

- We can also insert the string right into the `input` function call:

  ```
  x = input("Enter a new number ")
  x = float(x)
  print('The square of', x, 'is', x*x)
  ```

- A similar function exists to convert a string to an integer:

  ```
  x = input("Enter an integer ")
  x = int(x)
  ```

- We will use this idea to modify our area and volume calculation so that the user of the program types in the numbers.
  - The result is more useful and feels more like a real program (run from the command line).
  - It will be posted on the course website.

4.15 Practice Problems - Part 2

1. What is the output for this Python program?

   ```
   print(len('George'))
   print(len(' Tom  '))
   s = """Hi
   sis!
   """
   print(len(s))
   ```

2. Which of the following are legal? For those that are, show what Python outputs when these are typed directly into the interpreter.

   ```
   >>> 'abc' + str(5)
   >>> 'abc' * str(5)
   >>> 'abc' + 5
   >>> 'abc' * 5
   >>> 'abc' + 5.0
   >>> 'abc' + float(5.0)
   >>> str(3.0) * 3
   ```

3. What is the output of the following when the user types 4 when running the following Python program?

```
x = input('Enter an integer ==> ')
x = x*2
x = int(x)
x *= 2
print("x is:", x)
```

4. What is the output when the user types the value 64 when running the following Python program?

```
x = input('Enter an integer ==> ')
y = x // 10
z = y % 10
print(x, ',', y, z, sep='')
```

   What happens when you do not have the call to the `int` function?

5. Write a program that requests an integer from the user as an input and stores in the variable `n`. The program should then print `n` 1's with 0's inbetween. For example if the user input the value 4 then the output should be

```
1010101
```

4.16 Summary

- Strings represent character sequences — our third Python type
- String operations include addition (concatenate) and replication
- Functions on strings may be used to determine length and to convert back and forth to integers and floats.
- Escape sequences change the meaning of special Python characters or make certain characters have special meaning.
- Some special characters of note: `\n` for new line, `\t` for tab. They are each preceded by `\`
- The `print()` function offers significant flexibility.
- We can read input using `input()`