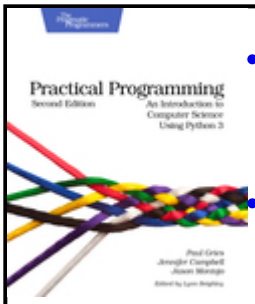
`str` Formatting



- Chapter 4.3 Creating a Multiline String
- Chapter 4.4 Printing Information
- Chapter 4.5 Getting Information from the Keyboard

Optional reading

Input/Output and `str` Formatting

Function `print`

Python has a built-in function named `print` that displays messages to the user. For example, the following function call displays the string `"hello"`:

```
>>> print("hello")
hello
```

In the output above, notice that `hello` is displayed without the quotation marks. The quotes are only for Python's internal string formatting and are not seen by the user.

The `print` function may also be called with a mathematical expression for an argument. Python evaluates the mathematical expression first and then displays the resulting value to the user. For example:

```
>>> print(3 + 7 - 3)
7
```

Finally, `print` can take in more than one argument. Each pair of arguments is separated by a comma and a space is inserted between them when they are displayed. For example:

```
>>> print("hello", "there")
hello there
```

`return` vs. `print`

Recall: The general form of a `return` statement:

`return` *`expression`*

When a `return` statement executes, the expression is evaluated to produce a memory address.

- *What is passed back to the caller?*
  That memory address is passed back to the caller.
- *What is displayed?*
  Nothing!

An example of `return`:

```
>>> def square_return(num):
    return num ** 2
>>> answer_return = square_return(4)
>>> answer_return
16
```

The general form of a `print` function call:

`print(`*`arguments`*`)`

When a `print` function call is executed, the argument(s) are evaluated to produce memory address(es).

- *What is passed back to the caller?*
  Nothing!
- *What is displayed?*
  The values at those memory address(es) are displayed on the screen.

An example of `print`:

```
>>> def square_print(num):
    print("The square of num is", num ** 2)
>>> answer_print = square_print(4)
The square num is 16
>>> answer_print
>>>
```

Function `input`

The function `input` is a built-in function that prompts the user to enter some input. The program waits for the user to enter the input, before executing the subsequent instructions. The value returned from this function is *always* a string. For example:

```
>>> input("What is your name? ")
What is your name? Jen
'Jen'
>>> name = input("What is your name? ")
What is your name? Jen
>>> name
'Jen'
>>> location = input("What is your location? ")
What is your location? Toronto
>>> location
'Toronto'
>>> print(name, "lives in", location)
Jen lives in Toronto
>>> num_coffee = input("How many cups of coffee? ")
How many cups of coffee? 2
>>> num_coffee
'2'
```

Operations on strings

Operation	Description	Example	Output
`str1 + str2`	concatenate `str1` and `str1`	`print('ab' + 'c')`	`abc`
`str1 * int1`	concatenate `int1` copies of `str1`	`print('a' * 5)`	`aaaaa`
`int1 * str1`	concatenate `int1` copies of `str1`	`print(4 * 'bc')`	`bcbcbcbc`

Triple-quoted strings

We have used single- and double- quotes to represent strings. The third string format uses triple-quotes and a triple-quoted string can span multiple lines. For example:

```
>>> print(''' How
are
you?''')
How
```

```
are
you?
```

Escape Sequences

Python has a special character called an *escape character*: `\`. When the escape character is used in a string, the character following the escape character is treated differently from normal. The escape character together with the character that follows it is an *escape sequence*. The table below contains some of Python's commonly used escape sequences.

Escape Sequence	Name	Example	Output

`\n`	newline (ASCII linefeed - LF)	`print('''How`
		`are`
		`you?''')`
			`How`
			`are`
			`you?`

`\t`	tab (ASCII horizontal tab - TAB)	`print('3\t4\t5')`	`3  4  5`
`\\`	backslash (\)	`print('\\')`	`\`
`\'`	single quote (')	`print('don\'t')`	`don't`
`\"`	double quote (")	`print("He says, \"hi\".")`	`He says, "hi".`

Jennifer Campbell • Paul Gries
University of Toronto