

Chapter 3 — Strings, Input, and Interactive Programs

Introduction

This chapter will go into a detailed discussion of strings, basic operations that you can do strings, asking questions, and using the answers to create interactive programs.

A string is a group of letters, numbers, and symbols surrounded by single or double quotes. The computer does not understand or evaluate the contents of a string, it just treats it as a chunk of stuff. We saw string literals surrounded by quotes and triple quotes in Chapter 1.

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Use the string operators to concatenate and repeat strings to build new strings.
- Apply the correct type of quote marks around a string.
- Show the length of a string.
- Convert strings to numbers and numbers to strings.
- Use input to ask the user for a value to be used in the program.
- Employ various string methods to format string output, locate substrings, and replace parts of a string with a new string.
- Apply the index operator to extract a single character or group of characters from a string.

Prerequisites

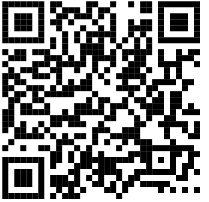
This chapter requires many of the concepts from Chapter 1. Material from Chapter 2 is not strictly required to continue with this chapter.

String Operations

- **+** — Concatenate (Add)
 - The order of strings is important (unlike with numeric addition)
 - If both are numbers they will be added.
 - If both are strings they will be concatenated.
 - If they are mixed types, an error will occur



- `"how " + "now"` evaluates to `how now`
- `"abcdef" + "ghij"` evaluates to `abcdefghij`
- `"abc'def" + 'ghi"jkl'` evaluates to `abc'defghi"jlk`
- *** — Repeat (Multiply)**
 - If both are numbers then multiply
 - if first is a string and second is an integer then repeat the string that many times. if n is ≤ 0 then return the empty string.
 - `"a"*10` evaluates to `aaaaaaaaaa`
 - `"cow"*-4` evaluates to `"` (the empty string)

+ *	String Operators
<p>The plus and multiply operators can also be used for strings. The concatenate operation, which appends two strings together, is performed by the plus. The multiply operator use a second integer argument to repeat a string the desired number of times.</p> <p>https://docs.python.org/3.7/library/stdtypes.html#common-sequence-operations http://bit.ly/2V8ILOS</p>	

Please support this work at


```
1| print("a" + "b")
2| a = "x"
3| a = a + "y"
4| print(a)
5| print("***3)
6| print("***2)
7| print("***1)
```

```
ab
xy
***
**
*
```

String Length Function

The `len()` function returns the length of a string, list, tuple, or a map. At this point we will use `len()` to tell us how long a string is.




<code>len(expression)</code>	Function
<code>len()</code> — returns the length of a string, list, tuple, or a map. https://docs.python.org/3/library/functions.html#len http://bit.ly/2ENddrZ	

```
1 | name = "El. Supremo"  
2 | print("Your name is", len(name), "letters long.")
```

```
Your name is 10 letters long.
```

Converting from a string to a number and back

The `float()` and `int()` functions will actually try to convert this expression or type into a float or an integer. If the expression can't be converted, an error will be thrown. See discussion in Chapter 4 for how to trap the error and continue the program.

<code>int(expression)</code>	Function
The <code>int()</code> function will convert a string or a floating-point number to an integer. If it is unable to make the conversion an error will be thrown. https://docs.python.org/3/library/functions.html#int http://bit.ly/js3int	

<code>float(expression)</code>	Function
--------------------------------	----------



The `float()` function will convert an integer or a string to a floating-point number. If it is unable to make the conversion, an error will be thrown.

<https://docs.python.org/3/library/functions.html#float>
<http://bit.ly/py3float>



```
1| print ( 1 + 2 )
2| print( int("1") + int(2.3))
3| q = float("4.56")
4| print(q * 3.9)
```

```
3
3
17.784
```

NOTE: The `int()` function will not convert a string with a decimal point ("1.23", "-655.36"). An error will be thrown. To convert a string that might contain a decimal point, it is recommended that you convert to a float first, then an integer. For example: `a= int(float("5.67"))`.

Be careful. A fatal error will occur if you try to convert a value that is not a number. The program will actually just quit when this happens. You will learn, later, how to handle errors.

```
1| value = float("fool")
2| print(value)
```

```
Traceback (most recent call last):
  File "/float_foo.py", line 1, in <module>
    value = float("fool")
ValueError: could not convert string to float: 'fool'
```

```
1| print(int("hello"))
```

```
Traceback (most recent call last):
  File "/int_hello.py", line 1, in <module>
    print(int('hello'))
ValueError: invalid literal for int() with base 10: 'hello'
```



The important part of the error message is shown in bold. The line number represents the line in your program where the problem happened and the last line is a detailed message about the error. See Appendix A for common errors and a description of what to look for to fix them.


The `str()` function works just like the `int()` and `float()` functions, except it will make a number or other objects into a string. If you want to contaminate numbers to strings, you must first convert them to strings.

<code>str(expression)</code>	Function
The <code>str()</code> function will convert an expression into a string. https://docs.python.org/3/library/functions.html#func-str http://bit.ly/2RlItGg	

```
1 | message = "Four score and " + str(7) + " years ago, ..."  
2 | print(message)  
  
Four score and 7 years ago, ...
```

Input a String

The `input()` function will ask the user to enter a value and the value entered will be returned as a string. You may also include a prompt message to the `input()` function.

<code>input()</code> <code>input(expression)</code>	Function
The <code>input()</code> function will accept user input and return the string value entered. You may optionally specify a string expression that will be printed to prompt the input. https://docs.python.org/3/library/functions.html#input	



<http://bit.ly/2AdEu3O>

```
1 | name = input("Enter your name")
2 | birth_year = input("What year were you born in?")
3 | age = 2017 - int(birth_year)
4 | print("Wow", name, "you are probably", age, "years old.");
```

If the user was named Jim and was born in 1984 the output would look like:

```
Wow Jim you are probably 33 years old.
```

Selected String Methods

This section introduces some methods that are available to manipulate strings. Python has many more but these will give you a start. Don't be afraid of the documentation to find others.

We have seen functions, like `print()` and `len()` that are built into the language. There are also a special type of functions that are called methods. Methods are part of the "object" model, where objects have properties (values associated to them) and methods (actions that can be done to them). In Python even simple values like integers, floats, and strings are represented as objects (see Chapter 9). Methods of an object are called by placing a dot after the value/variable followed by the method name with parentheses.

Upper and Lower

In the first example of string methods, we have the `.upper()` and `.lower()` methods. These methods do exactly what you think that they would do, return a new string that is all uppercase letters or a string that is all lowercase letters.

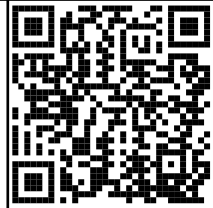
```
string.upper()  
string.lower()
```

Methods of String



The upper and lower methods of a string turn all of the letters to upper case or lower case.

<https://docs.python.org/3/library/stdtypes.html#index-30>
<http://bit.ly/2Q5ouGl>



```
1 | print("mellow".upper())
2 | color = "YeLLow"
3 | print(color.lower())
```

```
MELLOW
yellow
```

Strip, Lstrip, and Rstrip

Another really useful method is `.strip()`. It returns a string with leading and trailing white-space removed from the string. White-space characters usually include: Space (0x20), Tab (0x09), Line Feed (0x0a), Carriage Return (0x0d), Vertical Tab (0x0b), and Form Feed (0x0c). You may also tell strip which letters you want to have removed if you pass them as a string to the method.

```
string.strip()
string.lstrip()
string.rstrip()
```

Methods of String

The lstrip, rstrip, and strip methods remove white-space from the left, right, or both ends of a string. White-space includes: spaces, tabs, returns, and other forms of spacing.

<https://docs.python.org/3/library/stdtypes.html#index-30>
<http://bit.ly/2Q5ouGl>



```
1 | two = "  two  "
2 | two = two.strip()
3 | print("  one".strip() + two)
```



```
4 | print("00000099".lstrip("0"))
```

```
onetwo  
99
```

Count and Find

The next two string methods are `.count()` and `.find()`. They do exactly what their names imply.

The first, `.count()`, returns the number of occurrences that the "needle" is found in the "haystack". If the sub-string is not found then a zero will be returned.

The `.find()` method tells you at what position in the string the "needle" is found. If `.find()` does not find the "needle" in the string then a -1 is returned.

```
string.count(find_exp)  
string.count(find_exp, start_exp)  
string.count(find_exp, start_exp, end_exp)
```

Method of String

Return the count of the expression in the string. You may also specific starting and ending characters if you do not want to search the whole string.

<https://docs.python.org/3/library/stdtypes.html#index-30>
<http://bit.ly/2Q5ouGl>



```
string.find(find_exp)  
string.find(find_exp, start_exp)  
string.find(find_exp, start_exp, end_exp)
```

Method of String

Return the location of the expression in the string. If it is not found then return a -1. You may also specific starting and ending locations if you do not want to search the whole string.

<https://docs.python.org/3/library/stdtypes.html#index-30>
<http://bit.ly/2Q5ouGl>



NOTE: Character positions in a string are "zero based". This means that the first letter is at position 0 and the last is at `len()-1`. This may cause some confusion at first, you will also see "zero-based" indexes used throughout Python and other languages.

```
1| lyric = "we all live in a yellow submarine"
2| print("there are", lyric.count("ll"), "'ll' in the lyric")
3| print("the first one is at", lyric.find("ll"))
```

```
there are 2 'll' in the lyric
the first one is at 4
```

Replace

The `.replace()` method will replace one sub-string with another a specified number of times. If you do not specify how many replacements to make then all the occurrences will be replaced.

```
string.replace(old_exp, new_exp)
string.replace(old_exp, new_exp, n_exp)
```

Method of String

Find all occurrences of old string and replace the with the new one. If n is specified than only do replace a specific number of times.

<https://docs.python.org/3/library/stdtypes.html#index-30>
<http://bit.ly/2Q5ouGl>



```
1| lyric = "we all live in a yellow submarine"
2| lyric = lyric.replace("yellow", "pink")
3| print(lyric)
4| lyric = lyric.replace("all", "don't")
5| print(lyric)
```

```
we all live in a pink submarine
we don't live in a pink submarine
```



Ljust, Rjust, and Zfill

The next three methods will pad a string out with spaces, a specified character, or with zeros.

```
string.ljust(width_exp)
string.ljust(width_exp, fill_exp)
string.rjust(width_exp)
string.rjust(width_exp, fill_exp)
string.zfill(width_exp)
```

Methods of String

Left or right justify the string to the specified width. Space is used by default but you may optionally define a different pad character. The zfill method right justifies to a specific width with zeros to fill.



<https://docs.python.org/3/library/stdtypes.html#index-30>
<http://bit.ly/2Q5ouGI>

```
1 | print("name".ljust(20), "Id".rjust(5), "salary".rjust(8))
2 | print("Jane Doe".ljust(20), "101".zfill(5), "123456".rjust(8))
3 | print("Jo Cho".ljust(20), "386".zfill(5), "56789".rjust(8))
4 | print("Sam Suza".ljust(20), "222".zfill(5), "44444".rjust(8))
```

```
name                Id    salary
Jane Doe            00101 123456
Jo Cho              00386  56789
Sam Suza            00222  44444
```

Individual Characters and Slicing Strings

Python has a special operator, called the **indexing operator**, that will allow you to extract a single character, or sub-string from a string. The indexing operator is also used to access elements and sub-groups of elements from lists, tuples, and maps (Chapter 3).

The first use of the indexing operator ([] square brackets) is to extract individual characters from a string.

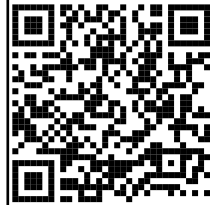


`string[pos_exp]`

String Operator

The subscript operator to extract the character at position `n`, where `n` is 0 to `len()-1`. If `n < 0` then extract a character from the right end of the string.

<https://docs.python.org/3/library/stdtypes.html#index-19>
<http://bit.ly/2CyCLaF>



```
1| alpha = "abcdefghijklmnopqrstuvwxy"
2| print("the third letter of the alphabet is", alpha[2])
3| print("the last letter of the alphabet is", alpha[-1])
4| n = 13
5| print("the",n,"letter of the alphabet is", alpha[n-1])
```

```
the third letter of the alphabet is c
the last letter of the alphabet is z
the 13 letter of the alphabet is m
```

`string[start_exp: end_exp]`

String Operator

The slice operator (`[:]` square brackets) is used to extract groups of characters and returns a string. The sub-string begins at position `start` and ends at `end-1`. If `start` is omitted begin at character 0, if `end` is omitted then include to end of string.

<https://docs.python.org/3/library/stdtypes.html#index-19>
<http://bit.ly/2CyCLaF>



```
1| part_number="001HG9IFN"
2| catalog = part_number[:3]
3| supplier = part_number[3:5]
4| group = part_number[5]
5| code = part_number[6:]
6| print("Item", part_number, "is found in catalog", catalog,',')
7| print("we buy it from supplier", supplier)
8| print("and internally use", group+'-'+code, "for short.")
```

```
Item 001HG9IFN is found in catalog 001 ,
```



```
we buy it from supplier HG  
and internally use 9-IFN for short.
```

NOTE: With strings `X[n]` is the same as `X[n:n+1]`. It is not recommended that you use the second syntax. With strings they return the same value, with other types this is not the case.

Summary

Goes here

Important Terms

- concatenate
- count
- find
- float
- index
- input
- int
- justify
- len
- ljust
- lower
- lstrip
- repeat
- replace
- rjust
- rstrip
- slice
- str
- strip
- substring
- upper
- zfill

Exercises

1. Write a python program that will have two variables `a` and `b` containing integers. Use string justification to display the sum of those two numbers on a "pretty" format. Test your program with integers of several lengths. See the example below.

```
      989823  
+      34  
-----  
      989857
```

2. Modify #1 to ask the user for the two numbers.

3. Write a python program to calculate loan payments. Ask the user for the interest rate (APR), the number of months the loan is for, and the amount to borrow. Convert the entries to floating-point numbers and use the in the formula below. Remember to convert APR to monthly interest rate by dividing by 100 and again by 12.



$$Payment = \frac{rate_{monthly} * principle}{1 - (1 + rate_{monthly})^{-months}}$$

```
Interest (APR) ?8
Months?24
Principal?2500
113.06822864046143
```

4. Your friend needs you to get in their locker because they forgot an important book. They have hidden the combination in the string below. Assign a variable to the string and then print out the count of the letters 'a', 'b', and 'c'. These are the three numbers to open the lock.

```
aaaacbccbacabbabbcaaccccaaccccbccab
```

5. Create a Python program that will change a string, that the user enters into your program, into l337. Translation to l337 (pronounced "leet") is done by changing some letters to numbers. A simplified conversion is shown in the table below.

i	1
e	3
a	4
t	7

6. Create a silly story generator that will ask the user for 5 words and then print a short story with those words. Use five input statements and variables to get and store the words. Use them in your program to build and display the following story.

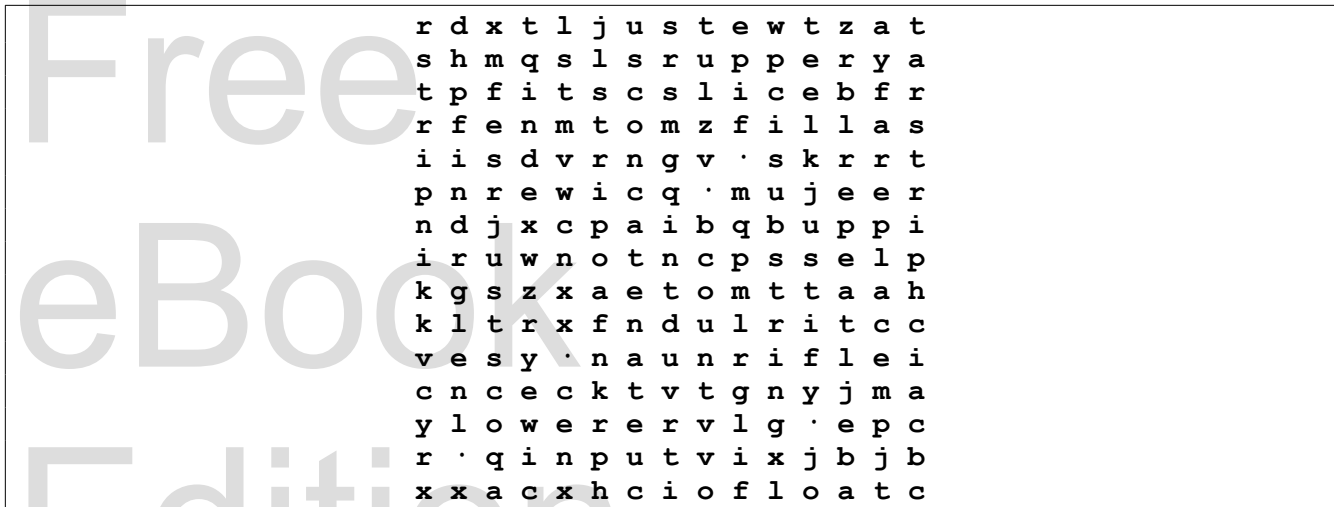
- A – An animal
- B – A name
- C – A food
- D – A period of time
- E – Another animal

There was a **A** named **B** who loved to eat **C**.
One **D** the **C** tasted so good that **A** turned into a **E**.

7. Create a program that asks for a person's first name and their last name. Extract the first letter and display their initials.



Word Search



concatenate, count, find, float, index, input, int, justify, len, ljust, lower, lstrip, repeat, replace, rjust, rstrip, slice, str, strip, substring, upper, zfill

References

<https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>

