Code Academy

Chapter Two

PROGRAMMING WITH NUMBERS AND STRINGS

Sections 2.4
Strings

Code Academy

Lecture Goals

- To learn how to use Python strings

2.4 Strings

Code Academy

Strings

- Definitions:
  - Text consists of **characters**
  - **Characters** are letters, numbers, punctuation marks, spaces, ….
  - A **string** is a sequence of **characters**
- In Python, string literals are specified by enclosing a sequence of **characters** within a matching pair of either single or double quotes.

```
print("This is a string.", 'So is this.')
```

- By allowing both types of delimiters, Python makes it easy to include an apostrophe or quotation mark within a string.
  - `message = 'He said "Hello"'`
  - Remember to use matching pairs of quotes, single with single, double with double

String Length

Code Academy

- The number of characters in a string is called the length of the string. (For example, the length of **"Harry"** is 5).
- You can compute the length of a string using Python's **len()** function:
-
```
length = len("World!")     # length is 6
```
- A string of length 0 is called the empty string. It contains no characters and is written as "" or ''.

Examples: What is the length of the string "Python Program"?

14

P	y	t	h	o	n		P	r	o	g	r	a	m
1	2	3	4	5	6	7	8	9	10	11	12	13	14

String Concatenation ("+")

- You can ‘add’ one String onto the end of another

```
firstName = "Harry"
lastName = "Morgan”
name  = firstName + lastName  # HarryMorgan
print(“my name is:”, name)
```

- If you want a space in between the two names then

```
name = firstName + " " + lastName # Harry Morgan
```

- Using “+” to concatenate strings is an example of a concept called operator *overloading*. The “+” operator performs different functions of variables of different types

String repetition (“*”)

- You can also produce a string that is the result of repeating a string multiple times.
- Suppose you need to print a dashed line.

  - Instead of specifying a literal string with 50 dashes, you can use the * operator to create a string that is comprised of the string "-" repeated 50 times.

```
dashes = "-" * 50
```

  - results in the string:

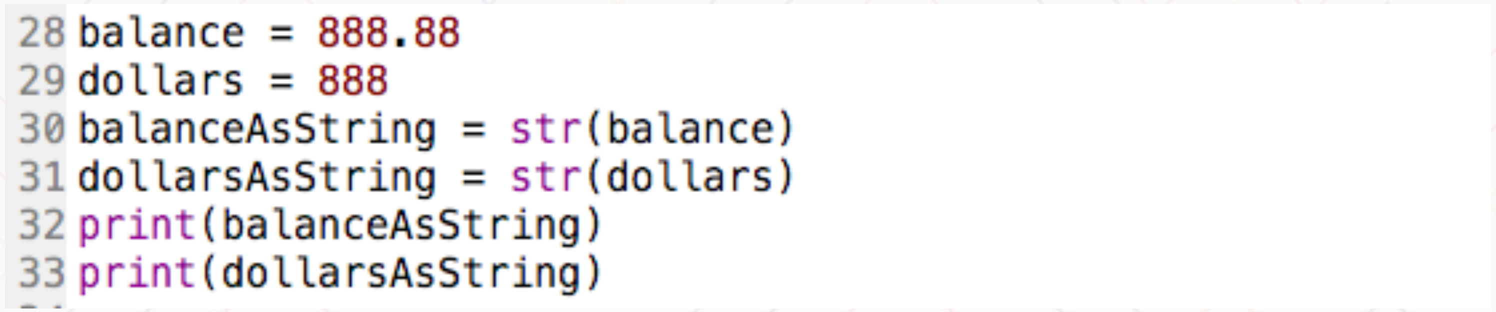
```
"-------------------------------------------------”
```

- The “*” operator is also *overloaded*.

Converting Numbers to Strings

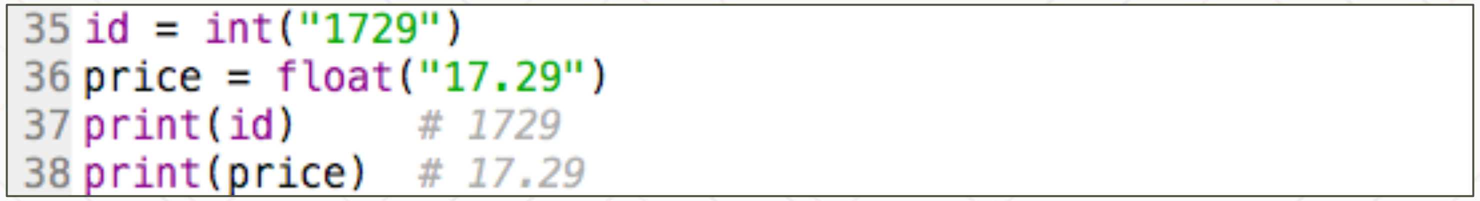
- Use the **str()** function to convert between numbers and strings.

  - Example:



```
balance = 888.88
dollars = 888
balanceAsString = str(balance)
dollarsAsString = str(dollars)
print(balanceAsString)
print(dollarsAsString)
```

- To turn a string containing a number into a numerical value, we use the `int()` and `float()` functions:



```
id = int("1729")
price = float("17.29")
print(id)       # 1729
print(price)    # 17.29
```

- This conversion is important when the strings come from user input

Code Academy

Strings and Characters

- **strings** are sequences of **characters**
- Python uses **Unicode** characters
  - **- Unicode** defines over 100,000 characters
  - **- Unicode** was designed to be able to encode text in essentially all written languages

- Characters are stored as integer values
  - See the ASCII (*the American Standard Code for Information Interchange*) subset on Unicode chart in Appendix D
  - For example, the letter 'C' has a value of 67

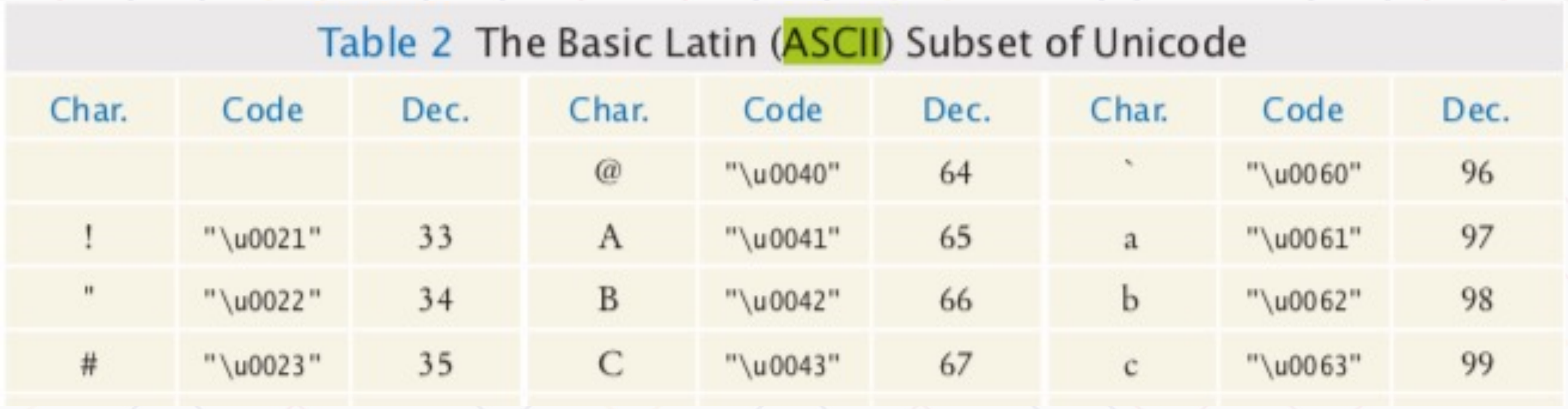


Table 2 The Basic Latin (ASCII) Subset of Unicode

Char.	Code	Dec.	Char.	Code	Dec.	Char.	Code	Dec.
			@	"\u0040"	64	`	"\u0060"	96
!	"\u0021"	33	A	"\u0041"	65	a	"\u0061"	97
"	"\u0022"	34	B	"\u0042"	66	b	"\u0062"	98
#	"\u0023"	35	C	"\u0043"	67	c	"\u0063"	99

Copying a character from a String

- Each character inside a String has an index number:

0	1	2	3	4	5	6	7	8	9
c	h	a	r	s		h	e	r	e

- The *first character* is in index zero (0)
- The [ ] operator returns a character at a given index inside a String:

0	1	2	3	4
H	a	r	r	y

```
name = "Harry"
start = name[0]     # "H"
last = name[4]     # "y"
```

- The index value must be within the valid range of character position. If not it will produce an Index error: string index out of range.

Code Academy

String Operations

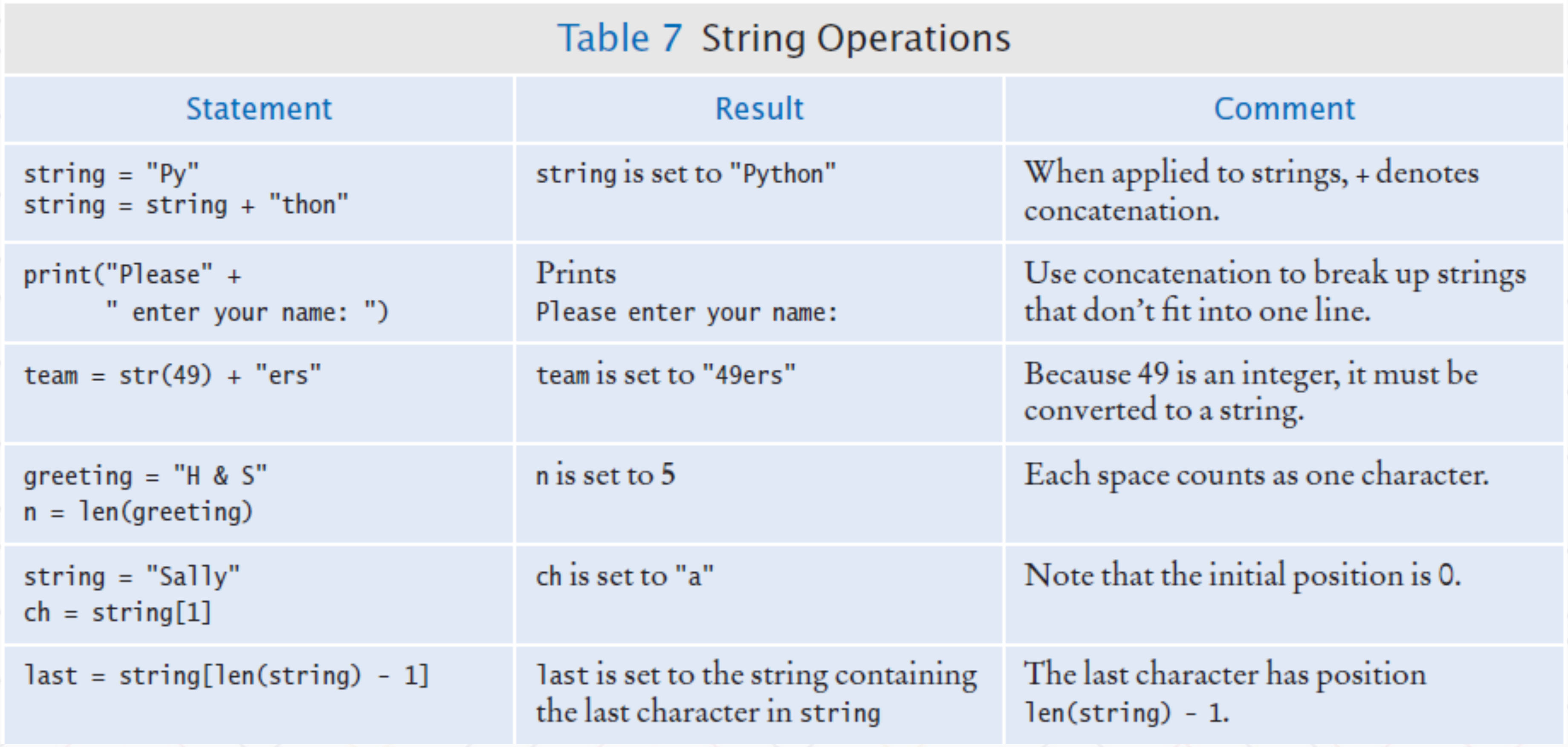


Table 7 String Operations

Statement	Result	Comment

`string = "Py"`
`string = string + "thon"`
	`string` is set to `"Python"`	When applied to strings, `+` denotes concatenation.

`print("Please" +`
`      " enter your name: ")`
	Prints
	`Please enter your name:`
		Use concatenation to break up strings that don't fit into one line.

`team = str(49) + "ers"`	`team` is set to `"49ers"`	Because `49` is an integer, it must be converted to a string.

`greeting = "H & S"`
`n = len(greeting)`
	`n` is set to 5	Each space counts as one character.

`string = "Sally"`
`ch = string[1]`
	`ch` is set to `"a"`	Note that the initial position is `0`.

`last = string[len(string) - 1]`	`last` is set to the string containing the last character in `string`	The last character has position `len(string) - 1`.

String Escape Sequences

- How would you print a double quote?
  - Preface the " with a “\” inside the double quoted String

```
print("He said \"Hello\"")
```

- OK, then how do you print a backslash?
  - Preface the \ with another \

```
print("C:\\Temp\\Secret.txt")
```

- Special characters inside Strings
  - Output a newline with a ‘\n’

```
print("*\n**\n***\n")
```

```
*
**
***
```

Code Academy

Summary: Strings

- Strings are sequences of characters.
- The `len()` function yields the number of characters in a String.
- Use the **+** operator to concatenate Strings; that is, to put them together to yield a longer String.
- In order to perform a concatenation, the + operator requires both arguments to be strings. Numbers must be converted to strings using the `str()` function.
- String index numbers are counted starting with 0.
- Use the `[ ]` operator to extract the elements of a String.