TECHNION
Azrieli Continuing Education and External Studies Division

Module 5.1.3: Booleans, None and Strings

Booleans

Booleans

- A Boolean can either be 1 or 0.
  - True (1)
  - False (0)

- In programming, we use Boolean evaluations to determine whether an expression is True or False.

- You can evaluate any expression in Python and get one of the two answers.

- A Boolean type belongs to the ‘bool’ class.

Boolean Values of Variables

- Any “empty” variables are False.
- Any variables with “content” are True.

```
>>> bool(13)
True
>>> bool(0)
False
>>> bool("Text")
True
>>> bool("")
False
```

Boolean Expressions

- With the use of comparison operators, we can create Boolean expressions which are expressions that evaluate True or False based on the current state of variables?

- Comparison operators look at variables but **do not** change the variables.

Python	Meaning
<	Less than
<=	Less or equal
==	Equal
>=	Greater or equal
>	Greater than
!=	Not equal

None Data Type

None Data Type

- A None Data Type is a single object to say – “No Value”!
- It represents nothing.
- Do not mistake it with a 0 / False!

Strings

Strings

- A String Data Type is a sequence of characters.
- Each character is a single letter.

String	SOME TEXT								
Char	S	O	M	E		T	E	X	T

Defining Strings

- To define a String in Python, use either single quotes or double-quotes.
- Python interpreter treats them the same!
- That means we can use both options to define a string; the only difference is how we, as programmers decide to use them.

```
>>> string1 = "Text"
>>> string2 = 'Text'
>>> str_w_qoutes1 = "Text with 'quotes'"
```

Escape Characters

- There is a unique string in built-in the language, the backslash!
- With the use of the ”escape string,” we signal the interpreter that we want to ignore the next character!
- An escaped character is one character long!
- It comes after the backslash!

Escape Characters

Character	Name	Syntax
‘	Single quote	\’
“	Double quote	\”
\	Back slash	\\
n	New line	\n
t	Tab	\t
r	Carriage return	\r

Raw Strings

- Using Raw Strings, Python ignores all backslashes, not treating them as escape characters anymore!

- To define a raw string, we add ‘r’ before defining our string.

```
>>> file_path = 'c:\temp\newfile.txt'
>>> print(file_path)
c:      emp
ewfile.txt

>>> file_path = r'c:\temp\newfile.txt'
>>> print(file_path)
c:\temp\newfile.txt
```

Length of Strings

- When we want to know a string's length, we can use the built-in function à len()
- The len() function returns the length of a string for us.

```
>>> print(file_path)
c:\temp\newfile.txt
>>> len(file_path)
19
```

Indexing

- We can use square brackets – [] with the index of the wanted letter in them to find a single letter inside a string!
- The index is the place number of the letter.
- **The count always starts with 0.**

String	SOME TEXT								
Char	S	O	M	E		T	E	X	T
Index	0	1	2	3	4	5	6	7	8

Indexing Cont.

- We can index strings with negative numbers!
- In which case indexing occurs from the end of the string backward!

String	**SOME TEXT**								
Char	S	O	M	E		T	E	X	T
Index	0	1	2	3	4	5	6	7	8
Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

Strings Slicing

- String Slicing is a useful technique to master! With String Slicing, we can create a new string out of a string by using the index of a string. When not writing one of the values, Python uses their default:

- If, the first value is missing. The string will be sliced from the beginning.

- If the second value is missing; the string will be sliced to the end.

```
my_string[m:a]
          m -> Staring position
          a -> Up to but not including
```

Strings Slicing Cont.

- There is a third argument in String Slicing, which represents Increment/Decrement!
- When not writing a third value, Python uses their default → increment by one.

```
my_string[m:a:d]
          m -> Staring position
          a -> Up to but not including
          d -> Jump by
```

The in Argument

- The *in* operator is a Boolean Type Operator.

- We use the in operator to evaluate whether an operand is contained within another.

- The *in* operator returns True if the first operand is contained within the second, and False otherwise.

- There is also a *not* in operator, which does the opposite!

String Methods

Method Name	Explanation	Example
count	Counts the number of times a substring appears.	'the sun is the best'.count('the')
upper	Turns all letters into uppercase.	‘A miXEd StrinG’.upper()
lower	Turns all letters into lowercase.	‘aNOTher MiXeD stRING‘.lower()
replace	Replaces all occurrences of a substring with another substring.	'I thpeak in lithp'.replace('th', 's')
find	Finds the index of the first appearance of a substring. If none is found, -1 is returned.	'Hallelujah!'.find('jah')
isdigit	Returns True/False, depending on if the string is built of only digits (0-9)	‘911’.isdigit()

Thank You!