

Reference guide: Python concepts from module 3; Terms and definitions from Course 7, Module 3

Built-in functions

The following built-in functions are commonly used in Python.

str()

Converts the input object to a string

```
str(10)
```

Converts the integer 10 to the string "10"

len()

Returns the number of elements in an object

```
print(len("security"))
```

Returns and displays 8, the number of characters in the string "security"

String methods

The following methods can be applied to strings in Python.

.upper()

Returns a copy of the string in all uppercase letters

```
print("Security".upper())
```

Returns and displays a copy of the string "Security" as "SECURITY"

.lower()

Returns a copy of the string in all lowercase letters

```
print("Security".lower())
```

Returns and displays a copy of the string "Security" as "security"

.index()

Finds the first occurrence of the input in a string and returns its location

```
print("Security".index("c"))
```

Finds the first occurrence of the character "c" in the string "Security" and returns and displays its index of 2

List methods

The following methods can be applied to lists in Python.

.insert()

Adds an element in a specific position inside the list

```
username_list = ["elarson", "fgarcia", "tshah"]  
username_list.insert(2, "wjaffrey")
```

Adds the element "wjaffrey" at index 2 to the username_list; the list becomes ["elarson", "fgarcia", "wjaffrey", "tshah"]

.remove()

Removes the first occurrence of a specific element inside a list

```
username_list = ["elarson", "bmoreno", "wjaffrey", "tshah"]  
username_list.remove("elarson")
```

Removes the element "elarson" from the username_list; the list becomes ["fgarcia", "wjaffrey", "tshah"]

.append()

Adds input to the end of a list

```
username_list = ["bmoreno", "wjaffrey", "tshah"]
username_list.append("btang")
```

Adds the element "btang" to the end of the username_list; the list becomes ["fgarcia", "wjaffrey", "tshah", "btang"]

.index()

Finds the first occurrence of an element in a list and returns its index

```
username_list = ["bmoreno", "wjaffrey", "tshah", "btang"]
print(username_list.index("tshah"))
```

Finds the first occurrence of the element "tshah" in the username_list and returns and displays its index of 2

Additional syntax for working with strings and lists

The following syntax is useful when working with strings and lists.

+ (concatenation)

Combines two strings or lists together

```
device_id = "IT"+"nwp12"
```

Combines the string "IT" with the string "nwp12" and assigns the combined string of "ITnwp12" to the variable device_id

```
users = ["elarson", "bmoreno"] + ["tshah", "btang"]
```

Combines the list ["elarson", "bmoreno"] with the list ["tshah", "btang"] and assigns the combined list of ["elarson", "bmoreno", "tshah", "btang"] to the variable users

[] (bracket notation)

Uses indices to extract parts of a string or list

```
print("h32rb17"[0])
```

Extracts the character at index 0, which is ("h"), from the string "h32rb17"

```
print("h32rb17"[0:3])
```

Extracts the slice `[0:3]`, which is `("h32")`, from the string `"h32rb17"`; the first index in the slice (0) is included in the slice but the second index in the slice (3) is excluded

```
username_list = ["elarson", "fgarcia", "tshah"]
print(username_list[2])
```

Extracts the element at index 2, which is `("tshah")`, from the `username_list`

Regular expressions

The following `re` module function and regular expression symbols are useful when searching for patterns in strings.

`re.findall()`

Returns a list of matches to a regular expression

```
import re
re.findall("a53", "a53-32c .E")
```

Returns a list of matches to the regular expression pattern `"a53"` in the string `"a53-32c .E"`; returns the list `["a53"]`

`\w`

Matches with any alphanumeric character; also matches with the underscore (`_`)

```
import re
re.findall("\w", "a53-32c .E")
```

Returns a list of matches to the regular expression pattern `"\w"` in the string `"a53-32c .E"`; matches to any alphanumeric character and returns the list `["a", "5", "3", "3", "2", "c", "E"]`

•

Matches to all characters, including symbols

```
import re
re.findall(".", "a53-32c .E")
```

Returns a list of matches to the regular expression pattern `"."` in the string `"a53-32c .E"`; matches to all characters and returns the list `["a", "5", "3", "-", "3", "2", "c", " ", ".", "E"]`

`\d`

Matches to all single digits

```
import re
re.findall("\d", "a53-32c .E")
```

Returns a list of matches to the regular expression pattern "\d" in the string "a53-32c .E"; matches to all single digits and returns the list ["5", "3", "3", "2"]

\s

Matches to all single spaces

```
import re
re.findall("\d", "a53-32c .E")
```

Returns a list of matches to the regular expression pattern "\s" in the string "a53-32c .E"; matches to all single spaces and returns the list [" "]

\.

Matches to the period character

```
import re
re.findall("\.", "a53-32c .E")
```

Returns a list of matches to the regular expression pattern "\." in the string "a53-32c .E"; matches to all instances of the period character and returns the list [". "]

+

Represents one or more occurrences of a specific character

```
import re
re.findall("\w+", "a53-32c .E")
```

Returns a list of matches to the regular expression pattern "\w+" in the string "a53-32c .E"; matches to one or more occurrences of any alphanumeric character and returns the list ["a53", "32c", "E"]

Represents, zero, one or more occurrences of a specific character

```
import re
re.findall("\w*", "a53-32c .E")
```

Returns a list of matches to the regular expression pattern `"\w*"` in the string `"a53-32c.E"`; matches to one or more occurrences of any alphanumeric character and returns the list `["a53", " ", "32c", " ", " ", "E"]`

`{ }`

Represents a specified number of occurrences of a specific character; the number is specified within the curly brackets

```
import re
re.findall("\w{3}", "a53-32c.E")
```

Returns a list of matches to the regular expression pattern `"\w{3}"` in the string `"a53-32c.E"`; matches to exactly three occurrences of any alphanumeric character and returns the list `["a53", "32c"]`

Glossary terms from module 3

Algorithm: A set of rules that solve a problem

Bracket notation: The indices placed in square brackets

Debugging: The practice of identifying and fixing errors in code

Immutable: An object that cannot be changed after it is created and assigned a value

Index: A number assigned to every element in a sequence that indicates its position

List concatenation: The concept of combining two lists into one by placing the elements of the second list directly after the elements of the first list

List data: Data structure that consists of a collection of data in sequential form

Method: A function that belongs to a specific data type

Regular expression (regex): A sequence of characters that forms a pattern

String concatenation: The process of joining two strings together

String data: Data consisting of an ordered sequence of characters

Substring: A continuous sequence of characters within a string

Revision #2

Created 27 December 2023 11:42:25 by naruzkurai

Updated 27 December 2023 11:43:23 by naruzkurai