SKILLS NETWORK

Python Cheat Sheet: The Basics

Learn Python for Data Science, AI & Development on Coursera

Python Data Types

String

Series of characters or data stored as text

```
my_string = "Hello"
```

String Operations

```
# returns the string with all uppercase letters
my_string.upper()
```

```
# returns the length of a string
len(my_string)
```

```
# returns the index of the first instance of the string inside the
# subject string, otherwise -1
my_string.find('l')
```

```
# replaces any instance of the first string with the second in my_string
my_string.replace('H', 'C')
```

Integer

A whole number

```
my_integer = 12321
```

Float

A decimal number

```
my_decimal = 3.14
```

Boolean

Discrete value true or false

```
a = True
b = False
```

Dictionary

Changeable collection of key-value pairs

```
my_dictionary = {'banana': 1, 12: 'laptop', (0,0):'center'}
```

Dictionary Operations

```
# Access value using key
my_dictionary['banana']
```

```
# Get all keys in a dictionary as a list
my_dictionary.keys()
```

```
# Get all values in a dictionary as a list
my_dictionary.values()
```

Tuple

Unchangeable collection of objects

```
tup = (1, 3.12, False, "Hi")
```

List

Changeable collection of objects

```
my_collection = [1, 1, 3.12, False, "Hi"]
```

List Operations

```
# returns the length of a list
len(my_collection)
```

```
# Add multiple items to a list
my_collection.extend(["More", "Items"])
```

```
# Add a single item to a list
my_collection.append("Single")
```

```
# Delete the object of a list at a specified index
del(my_collection[2])
```

```
# Clone a list
clone = my_collection[:]
```

```
# Concatenate two lists
my_collection_2 = ["a", "b", "c"]
my_collection_3 = my_collection + my_collection_2
```

```
# Calculate the sum of a list of ints or floats
number_collection = [1,2,3,4.5]
sum(number_collection)
```

```
# Check if an item is in a list, returns Boolean
item in my_collection
# Check if an item is not in a list, returns Boolean
item not in my_collection
```

Set

Unordered collection of unique objects

```
a = {100, 3.12, False, "Bye"}
b = {100, 3.12, "Welcome"}
```

Set Operations

```
# Convert a list to a set
my_set = set([1,1,2,3])
```

```
# Add an item to the set
a.add(4)
```

```
# Remove an item from a set
a.remove("Bye")
```

```
# Returns set a minus b
a.difference(b)
```

```
# Returns intersection of set a and b
a.intersection(b)
```

```
# Returns the union of set a and b
a.union(b)
```

```
# Returns True if a is a subset of b, false otherwise
a.issubset(b)
```

```
# Returns True if a is a superset of b, false otherwise
a.issuperset(b)
```

Indexing

Accessing data from a string, list, or tuple using an element number

```
my_string[element_number]
my_collection[element_number]
my_tup[element_number]
```

Slicing

Accessing a subset of data from a string, list, or tuple using element numbers from start to stop -1

```
my_string[start:stop]
my_collection[start:stop]
my_tup[start:stop]
```

Comparison Operators

Comparison Operators compare operands and return a result of true or false

Equal

```
a == b
```

Less Than

```
a < b
```

Greater Than

```
a > b
```

Greater Than or Equal

```
a >= b
```

Less Than or Equal

```
a <= b
```

Not Equal

```
a != b
```

Python Operators

- +: Addition
- -: Subtraction
- *: Multiplication
- /: division
- //: Integer Division (Result rounded to the nearest integer)

Conditional Operators

Conditional Operators evaluate the operands and produce a true of false result

And - returns true if both statement a and b are true, otherwise false

```
a and b
```

Or - returns true if either statement a or b are true, otherwise false

```
a or b
```

Not - returns the opposite of the statement

```
not a
```


© Copyright IBM Corporation 2021. All rights reserved.