Python Essentials Cheat Sheet

by Redjumpman via cheatography.com/35253/cs/17793/

Strings		
str.split(sep)	Splits a string by the separator and returns a list of strings	"1,3,4,5".split(',') --> ["1", "2", "3"]
str.strip()	Removes leading and trailing whitespace	" hello ".strip() --> "hello"
str.replace(old, new)	Replaces all *old* with *new* in the string	"Hello Moon".replace("Moon", "World") --> "Hello World"
str.join(iterable)	concatenates a list of strings.	", ".join(["1", "2", "3"]) --> "1, 2, 3"
str.startswith(-string)	Returns True if str starts with *string*	"Hello".startswith("Hel") --> True
str.endswith(string)	Returns True if str endswith *string*	"Hello".endswith("o") --> True
str.lower()	Returns a string with all characters in lowercase	"ALLCAPS".lower() --> "allcaps"
str.upper()	Returns a string with all characters in uppercase	"loud noises".upper() --> "LOUD NOISES"
str.find(char)	Returns the first index in the string where char is found, otherwise returns -1.	"abracadbra".find("d") --> 6
str.count(char)	Returns an integer of the number of occurances of char in str.	"abracadbra".count("a") --> 4
str.format(*args, **kwargs)	Formats a string according to your specifiers	"The {} costs {} dollars.".format("movie", 10) --> "The movie costs 10 dollars."

Strings are immutable
You can concatenate strings together with the + operator
Strings are iterable and thus can be looped over.
Reverse a string with my_string[::-1]

Lists (Assume my_list = [1, 2, 3])		
list.append(element)	Appends an element to a list	my_list.append(4) --> [1, 2, 3, 4]
list.extend(sequence)	Extends a list by adding another list to the end	my_list.extend([4,5,6]) --> [1, 2, 3, 4, 5, 6]
list.pop(index)	Removes and returns an element in a list by index.	my_list.pop(1) --> 2
list.count(element)	Counts the occurences of element in the list	my_list.count(3) --> 1
list.index(element)	Returns the index of the element in the list	my_list.index(3) --> 2
list.clear()	Empties out a list	my_list.clear() --> []
list.remove(element)	Removes an element from a list	my_list.remove(1) --> [2, 3]
list.sort()	sorts a list	my_list.sort(reverse=True) --> [3, 2, 1]

Lists are mutable
You can access an element in a list via the index: my_list[index]
Update a value at a specific index with my_list[index] = "new"
Create a copy of a list with slicing: my_list[:]
Reverse a list with my_list[::-1]

By **Redjumpman**

cheatography.com/redjumpman/

Published 15th November, 2018.
Last updated 13th November, 2018.

Dictionaries (Assume d = {"a": 1, "b": 2, "c": 3})		
dict.copy()	Returns a **shallow** copy of the dictionary	copied = d.copy()
dict.values()	Returns a dictionary view of all the values	d.values() --> dict_view([1, 2, 3])
dict.items()		
dict.keys()	Returns a dictionary view of all the keys	d.keys() --> dict_view(['b', 'a', 'c'])
dict.get(key, default=None)	Attempts to return the value of the key specified, otherwise it returns the default	d.get("X", "not valid") --> "not valid"
dict.pop(key, default)	Pops a key. Returns a key error if default is not specified and the key dne.	d.pop('a') --> 1
dict.popitem()	Pops a random key value pair from the dictionary	d.popitem() --> ("b", 2)
dict.update(iterable_pair)	Updates the dictionary when given a iterable pairing.	d.update({"z": 4}) --> {"a": 1, "b": 2, "c": 3} d.update([('x', 3)]) --> {"a": 1, "b": 2, "c": 3, "x": 3}

Assume the keys in a dictionary will be unordered.

All keys **must** be hashable. If something is immutable (strings, tuples, etc), then it can be a key. Things like sets and lists cannot be keys.

Dictionaries are mutable
Access the value of a key with d[key]

File Operations		
f.close()	Closes an open file	f.close()
f.read(n)	reads up to n characters in the file, otherwise the entire file if n is not specified.	f.read() --> "Hello World\n"
f.readlines()	Returns a list of strings. Each line in the file is put in a list	f.readlines() --> ["Hello World\n", "Goodbye World-\n"]
f.write(content)	Writes the content to the file	f.write("Hello World")
f.writelines(iterable)	Writes a list of lines to the file	f.writelines(["hello world", "goodbye world"])
f.seek(offset, from)	Moves the current position of the file	f.seek(5) --> Move five characters ahead f.seek(-5, 0) --> Start at the first character and move back 5.
f.tell()	Returns the current position in the file, by counted characters	f.tell() --> 13
open(file_path, 'mode')	opens a file at the given file path	returns the raw, unread file.

Always close the file **RIGHT** after you read it. Not 50 lines later.
When using the with statement, you do not need to close the file.
readline() will return the first line. If you run it again, it returns the next line.
Each line ends in a "\n"
Reading a file always returns a string or a list of strings.

Sets (Assume s = {1, 2, 3, 4})		
s.isdisjoint(other)	Returns True if s and other do not share any elements.	s.isdisjoint({7, 9}) --> True
s.issubset(other)	Returns True if all elements of s is in other.	s.issubset({1, 2, 3, 4, 5}) --> True s <= {1, 2, 3, 4, 5} --> True
s.issuperset(other)	Return True if all elements other is in s.	s.issuperset({1, 4}) --> True s >= {1, 4} --> True
s.union(other)	Combines all unique elements	s.union({2, 3, 7}) --> {1, 2, 3, 4, 7} s | {2, 3, 7} --> {1, 2, 3, 4, 7}
s.intersection(other)	Set of elements common to both sets	s.intersection({1, 3, 5, 7}) --> {1, 3} s & {1, 3, 5} --> {1, 3}
s.difference(other)	Returns any elements that exists in s, but not in other.	s.difference({2, 3, 7}) --> {1, 4} s - {2, 3, 7} --> {1, 4}
s.symmetric_difference(other)	Returns all the elements that are not shared between the sets	s.symmetric_difference({2, 9}) --> {1, 3, 4, 9} s ^ {2, 9} --> {1, 3, 4, 9}
s.copy()	Returns a shallow copy of the set.	s.copy() --> {1, 2, 3, 4}

Frozen sets are immutable, but regular sets are mutable
Sets are **not** ordered
sets only contain unique values
Create sets with curly brackets --> {1, 2, 3}
Create empty sets **ONLY** with set(). an empty {} is an empty dictionary.

By **Redjumpman**

cheatography.com/redjumpman/

Published 15th November, 2018.
Last updated 13th November, 2018.