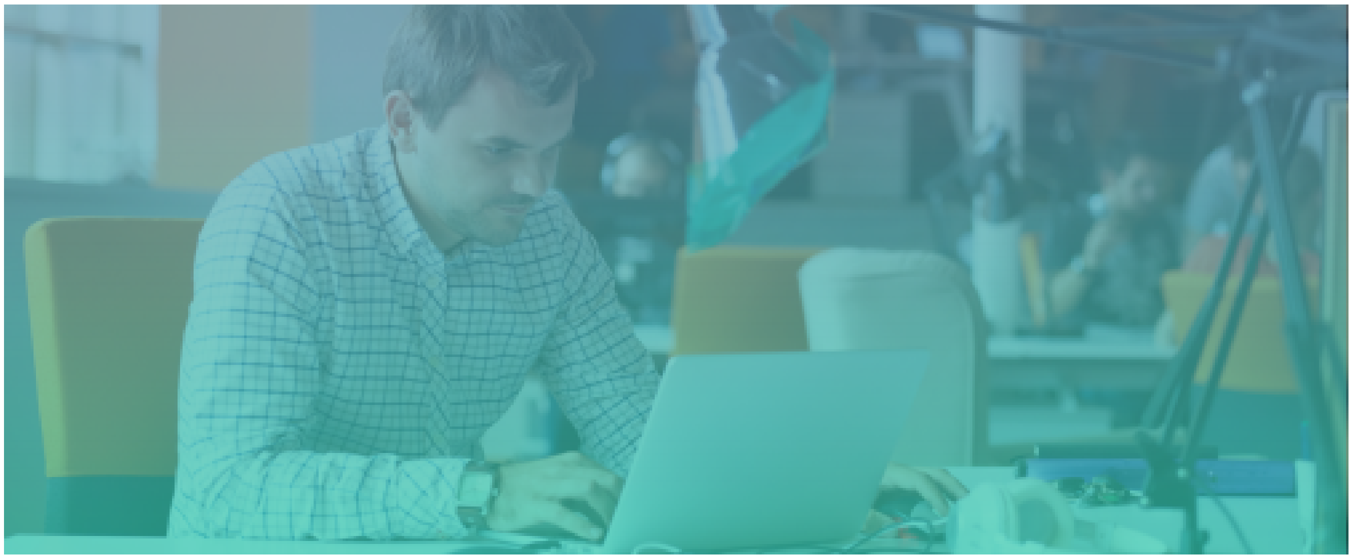
Python Cheat Sheet

How It Works

● This cheat sheet is a compilation of all of the syntax that is taught on the interactive coding environment platform.
● For each piece of syntax, you'll find a brief explanation, a code example, and the output of the example.
● This document is intended to be used as a quick reference when you can't remember a particular piece of syntax and it's blocking you from moving forward with your work. We encourage you to try to recall the syntax before you turn to this reference for help. In this way "truly master coding you will," as Master Yoda said (well, maybe he didn't say that exactly but you get the idea.)

Table of Contents



Syntax

Python Syntax

• print function:

Note the parentheses: ( ) and single quotes: ' '

Example:
print('Welcome to the DSC prep course!') print(10 + 34)

• Math in Python:

To add two values, use the + symbol.

Example: print(10 + 34)

```
Output: 44
```

To subtract two values, use the - symbol.

Example: print(10 - 41)

```
Output: -31
```

To divide two numbers, use the / symbol. Multiplication can be accomplished with the * symbol.

To perform the Exponent operation, use the ** symbol.

Example: print(2 ** 8)

```
Output: 256
```

To perform the Modulo operation, use the % symbol.

Example: print(13 % 4)

```
Output: 1
```

• Comments in Python:

```
# This is a comment
'''
This is a Python program!
Here's some more info about your program. '''
```

Control Flow - Part 1

• Booleans

Booleans is a special data type that can only have a value of True or False.

== / != (same / not the same)

> / < (greater than / less than)

>= / <= (greater than or equal to / less than or equal to)

Example:

```
one = 'Coding' == 'Coding'
two = 'Coding' == 'coding'
three = 1 >= 0
```

• If, elif, else

Note: If and Else can only be used once in each list or statement, but elif can be used multiple times — or not used at all.

Example:

```
animal = 'Bird'
if animal == 'Lion':
 print('Meat')
elif animal == 'Zebra':
 print('Grass')
```

```
else:
 print('Water')
```

```
Output: Water
```

- **for loops**

Example:

```
for i in range(4): #The range goes from 0 to 4 (not including 4).
 print(i)
```

```
Output:
0
1
2
3
```

- **Range:**

Example:

```
for i in range(3, 6): #The range goes from 3 to 6 (not including 6).
 print(i)
```

```
Output:
3
4
5
```

```
for i in range(0, 6, 2): #Increase the counter by a value  of 2 for each iteration.
 print(i)
```

```
Output:
0
2
4
6
```

Python List

- **Access list elements**

```
shopping_cart = ['eggs', 'banana', 'bread', 'butter']
print(shopping_cart[0])
```

```
Output: eggs
```

- **index()**

```
companies = ['Google', 'Amazon', 'Apple', 'Microsoft']
print(companies.index('Google')) #Find an index of an item in the list
```

```
Output: 0
```

- **List slicing:**

```
companies = ['Google', 'Amazon', 'Apple', 'Microsoft']
print(companies[1:3])
```

```
Output: ['Amazon', 'Apple']
```

```
print(companies[1:])
```

```
Output: ['Amazon', 'Apple', 'Microsoft']
```

```
print(companies[:2])
```

```
Output: ['Google', 'Amazon']
```

- **Change list data:**

```
companies[0] = 'LinkedIn'
print(companies)
```

```
Output: ['LinkedIn', 'Amazon', 'Apple', 'Microsoft']
```

- **Add list data:**

```
companies.append('YouTube')
print(companies)
```

```
Output: ['LinkedIn', 'Amazon', 'Apple', 'Microsoft', ‘YouTube']
```

- **Insert new data:**

```
companies = ['LinkedIn', 'Amazon', 'Apple', 'Microsoft', 'YouTube']
companies.insert(1, 'Google')
print(companies)
```

```
Output: ['LinkedIn', 'Google', 'Amazon', 'Apple', 'Microsoft', ‘YouTube']
```

• len() function

The len function tells the number of elements in a list.

```
companies =['LinkedIn', 'Google', 'Amazon', 'Apple',
'Microsoft', 'YouTube']
print(len(companies))
```

```
Output: 6
```

• pop() and remove() methods

The pop() method removes the elements at the position you specify; if you don't specify a position, it removes the last element in the list.

Example:

```
companies = ['LinkedIn', 'Google', 'Amazon', 'Apple',
'Microsoft', 'YouTube']
print(companies.pop(2))
```

```
Output: Amazon
```

The remove() method gets rid of the first occurrence of the element that you want to delete.

Example:

```
companies = ['LinkedIn', 'Google', 'Amazon', 'Apple',
'Microsoft', 'YouTube']
companies.remove('Amazon')
print(companies)
```

```
Output: ['LinkedIn', 'Google', 'Apple',
'Microsoft', 'YouTube']
```

• sum()

The sum() method returns the sum of all the elements in a list.

```
list_1 = [10, 30, 55]
print(sum(list_1))
```

```
Output: 95
```

• sorted()

The sorted() method sorts the elements of a given list in either an ascending or descending order.

```
list_2 = [40, 130, 55]
print(sorted(list_2))
```

```
Output: [40, 55, 130]
```

• max()

The max() method returns the largest element in a list.

```
list_2 = [40, 130, 55]
print(max(list_2))
```

```
Output: 130
```

Python Functions

• Function basics

```
def say_hello(name): # say_hello is the function and
name is the parameter
 print('Hello, ' + name + '!')
```

```
def concat_names(name1, name2): #this function takes
two parameters
 return name1 + ' ' + name2
```

• Call functions with data

```
def say_hello(first_name, last_name):
 name = first_name + ' ' + last_name
 print('Hello, ' + name + '!')
say_hello('Bobby', 'Jones')
```

```
Output: Hello, Bobby Jones!
```

• Call a function from another function

```
def user_name_function(f_name, l_name):
 return 'My name i ' + ' ' + f_name + ' ' + l_name
```

```
def user_greetings(f_name, l_name):
 print('Hello World!')
 print(user_name_function(f_name, l_name))
```

Python Dictionaries

• Dictionary structure

```
Dictionary_name = {key:value} #keys must be an
immutable type such as a string, number, or tuple
python_empty_dict = {} #dictionary definition first_dict =
{'banana': '1 kg', 'meat': '0.5 kg'} #dictionary  with strings as
its keys and values
second_dict = {'name': 'Petra', 1: [31, 43, 55]}
#dictionary with mixed keys
```

• get() method

```
person = {'name':'Mark', 'age': 16}
print(person.get('age')) #Accessing dictionary using
get() method
```

```
Output: 16
```

• Add new elements or change elements

my_dict = {'name':'Jack', 'age': 26}

# update value
my_dict['age'] = 27

#Output: {'age': 27, 'name': 'Jack'}
print(my_dict)

# add item
my_dict['address'] = 'Downtown'

• pop() method

Take a look at the pop() method and the popitem() method in action:

# create a dictionary
squares = {1:1, 2:4, 3:9, 4:16, 5:25}

# remove a particular item
print(squares.pop(4))

```
Output: 16
```

print(squares)

```
Output: {1: 1, 2: 4, 3: 9, 5: 25}
```

# remove an arbitrary item
print(squares.popitem())

```
Output: (1, 1)
```

print(squares)

```
Output: {2: 4, 3: 9, 5: 25}
```

• del() method

# create a dictionary
squares = {1:1, 2:4, 3:9, 4:16, 5:25}

# delete a particular item
del squares[5]
print(squares)

```
Output: {1:1, 2:4, 3:9, 4:16}
```

• clear() method

# create a dictionary
squares = {1:1, 2:4, 3:9, 4:16, 5:25}

# remove all items
squares.clear()

• key() method

my_dict = {1:1, 2:2, 3:3}

for key in my_dict.keys():
 print(my_dict[key])

• items()

for key, value in dict.items():
 print(key, values)

• Convert a dictionary to a list

list(dict.items())

• Convert a list to a dictionary

dict(list)

Strings

• String length

string_1 = "Hello World!"
print(len(string_1))

```
Output: 12
```

• Use index() to find a character

stirng_2 = "Python programming language"
print(string_2.index('o'))

```
Output: 4
```

• count()

string_3 = "Being a Data Scientist has provided great opportunities for me!"
print(string_3.count("a"))

```
Output: 5
```

print(string_3.count("Data"))

```
Output: 1
```

• String slicing

string_1 = "Hello world!"
print(string_1[:5])

```
Output: Hello
```

• upper() and lower()

```
string_4 = "Hello World!"
print(string_4.upper())
print(string_4.lower())
```

```
Output:
HELLO WORLD!
hello world!
```

• Split strings

```
string_5 = "Today is a very nice day!"
print(string_5.split(" ")) #split on space
```

```
Output: ['Today', 'is', 'a', 'very', 'nice',
'day!']
```

• startswith() and endswith()

```
string_6 = "Artificial Intelligence is cool!"
print(string_6.startswith("Artificial"))
print(string_6.endswith("nice!"))
```

```
Output:
True
False
```

Control Flow - Part 2

• Nested loops

Example:

```
for x in [1, 2, 3]:
 for y in [4, 5, 6]: #Do not use the same counter name  for
two loops
 print(x * y)
```

• break statement

Using a break statement terminates the loop that contains the statement.

Note: If the break statement is inside a nested loop, the break will terminate the innermost loop.

Example:

```
for char in 'world':
 if char == 'l':
 break
 print(char)
print('The end')
```

```
Output:
w
o
r
The end
```

• continue statement

The continue statement is used to skip the rest of the code —the code after the continue statement — inside a loop, but only for the current iteration. The loop continues to run from the next iteration and does not terminate.

Example:

```
for char in 'world':
 if char == 'l':
 continue
 print(char)

print('The end')
```

```
Output:
w
o
r
d
The end
```

• while loops:

Example:

```
word = 'snake'
usr_word = input('Type the word ' + word + ':') while
usr_word != word: #check that the word entered  matches
the requested word
 usr_word = input('Try again!: ')
print('Correct!')
```

Python Interacting with the Program

• Input program

Program input allows you to enter a value from the terminal that can then be used in your program.

Example:

```
name = input('Enter your name: ')
print('Hello, ' + name + '!')
```

• Create a text file and close it

```
file = open('new_file.txt', 'w') #Create a new, empty file
using the write mode
file.close()
```

• Write to a file

```
file = open('new_file.txt', 'w')
file.write('I love Python')
file.close()
```

• **read() method**

The output of the read() method will display all the text inside the file.

```
file = open('new_file.txt', 'r')
print(file.read())
file.close()
```

• **readline() method**

The readline() method will return a string of characters that contain a single line of text or information from the provided file.

```
file = open('new_file.txt', 'r')
print(file.readline())
file.close()
```

• **Appending to a file**

```
file = open('new_file.txt', 'a+') #If you want to append to a file, you need to open it in a+ mode, rather than the w mode.
file.write('Appended line 1')
file.write('Appended line 2')
file.close()
```

Stacks

• **LIFO**

The last item (be it a book, integer, string, etc.) added to a stack will be the first one removed if you try to access any other part of the stack. This kind of stack situation is known as last-in, first-out, or LIFO.

• **push()**

```
stack=[]
def push(stack,new_item): #add items to a stack by using the push() operation
 stack.append(new_item)
```

• **pop()**

Use the pop() method to remove the top item from the stack.

```
stack_example.pop()
```

• **is_empty()**

The is_empty() function takes a stack and checks to see if it contains any items.

```
def is_empty(stack):
 return stack==[]
```

• **size()**

Size() is a built-in function that will return the number of elements in a given stack.

Queues

• **FIFO**

Consider a queue that has formed at an ice cream stand. The first person who got into the queue is also the first person who will be served and leave the queue first. This way of entering and exiting (or adding and deleting) items from a queue is known as first-in, first-out, or FIFO.

```
queue_test = []
```

• **enqueue()**

```
def enqueue(queue,new_item): #add new items to a queue
 queue.append(new_item)
```

• **dequeue()**

Use dequeue to remove an item from a queue.

```
def dequeue(queue):
 queue.pop(0)
```