



Python 3 Cheat Sheet

[MLtrends](#)

Comments

```
1 # This is a one line comment
2
3 '''
4 This is a multi-line
5 comment
6 '''
7
8 """
9 This is a multi-line
10 comment too
11 """
```

String

```
1 str_1 = "Hello, World!" # Double quotes
2 str_2 = 'This is a string too' # Single quotes
3 str_3 = '''This is
4 a long string''' # Multi-line string
5 str_4 = "HI\nThere" # \n = Linefeed, start a new line
6 str_5 = "HI,\tThere" # \t = Add tab/indent
7 str_6 = "He\'s fine" # \' = Print a single quote ( ' ) in text
8 str_7 = "It is called \"IT\"" # \" = Print a single quote ( " ) in text
9 str_8 = "c:\\path\\files" # \\ Print a backslash ( \ ) in text
10 str_9 = r"\n means linefeed" # Treat ( \n ) as a literal character
11 str_10 = str(3) # Convert a number to string type
12 length = len(str_1) # Check the length of a string
```

String Concatenation

```
1 str_1 = "Hello"
2 str_2 = "World"
3
4 sentence = str_1 + ", " + str_2 + "!"
5 # Hello, World!
6
7 words = ["This", "is", "an", "amazing", "thing"]
8 print(' '.join(words))
9 # This is an amazing thing
10 print(', '.join(words))
11 # This, is, an, amazing, thing
```

Input/Output

```
1 str_1 = "Python 3"
2 str_2 = "Cheat sheet"
3
4 print("%s %s" % (str_1, str_2)) # Python 3 Cheat sheet
5 print("{} {}".format(str_1, str_2)) # Python 3 Cheat sheet
6 print("{0} {1}".format(str_1, str_2)) # Python 3 Cheat sheet
7 print("{1} {0}".format(str_1, str_2)) # Cheat sheet Python 3
8 print(f"{str_1} {str_2}") # Python 3 Cheat sheet
9
10 your_age = input("How old are you?") # The input is string type
11 print(f"{60-int(your_age)} years to retire") # Convert string to integer
```

Defining and Calling a Function

```
1 def display_something(): # Define a function
2     print("This is a demo!")
3
4 display_something() # Call the function
```

Function with Arguments

```
1 def my_sum1(x, y):
2     res = x + y
3     return res
4
5 my_sum1(3, 4) # Returns 7
6 my_sum1(y = 4, x = 3) # Return 7
7
8 # Add optional argument for the function
9 def my_sum2(x, opt = 0):
10     res = 2*x + opt
11     return res
12
13 my_sum2(3), my_sum2(3, opt = 4)
```

Function with Variable No. of Arguments

```
1 # *args for non-key worded, variable number of arguments
2 def my_args_fun(*args):
3     print(args)
4
5 my_args_fun("non", "key", "worded", "variable no. of arguments")
6
7 # *kwargs for variable number of keyword arguments
8 def my_kwargs_fun(**kwargs):
9     print(kwargs)
10
11 my_kwargs_fun(first = 32, mid = 1, last = 22)
12
13 # Using *args and **kwargs for a function
14 def my_fun(*args, **kwargs):
15     print("args: ", args)
16     print("kwargs: ", kwargs)
17
```

Lambda Function

```
1 import math
2
3 def my_sqrt(x): # define a regular function
4     return math.sqrt(x)
5
6 print(my_sqrt(4))
7
7 g = lambda x: math.sqrt(x) # define an online function
8 print(g(4))
```

Global Variable

```
1 counter = 0
2
3 def my_count():
4     global counter
5     counter = counter + 1
6     print(counter)
7
8 my_count()
```

Lists

```
1 str_list = ["hello", "world"]
2 mixed_list = [2, 3.1, "hi"]
3 list_of_list = [str_list, mixed_list]
4
5 print(str_list[0])
6 print(len(str_list))
7 print(len(mixed_list))
8 print(len(list_of_list))
```

List Methods

```
1 int_list = [3, 2, 5]
2 int_list.append(7) # Append a number to the list
3 print(int_list) # [3, 2, 5, 7]
4
5 int_list.extend([1, 6]) # Extend numbers to the list
6 print(int_list) # [3, 2, 5, 7, 1, 6]
7
8 print(int_list + [1, 6]) # Extend numbers to the list
9 # [3, 2, 5, 7, 1, 6, 1, 6]
10
11 int_list.pop() # Remove the last element from the list
12 print(int_list) # [3, 2, 5, 7, 1]
13
14 int_list.pop(3) # Remove the last element at index 3
15 print(int_list) # [3, 2, 5, 1]
16
17 int_list.remove(3) # Remove the first matching element
18 print(int_list) # [2, 5, 1]
19
20 int_list.insert(2, 8) # Insert 5 into index 2
21 print(int_list) # [2, 5, 8, 1]
```

List Unpacking

```
1 nums = [18, 88]
2 age, math = nums
3 print("The math is %d" % math) # The math is 88
4 age, _ = nums # Only interested in age
5 print(age) # 18
6
7 r = range(5) # Create a range from 0 to 4
8 print(*r) # Prints [0 1 2 3 4], unpack the range
9
10 print(r[0]) # Prints 0, number at 0 index
11 print(r[-1]) # Prints 4, number at the last index
12 print(r[-2]) # Prints 3, number at the second last index
```

Slicing

```
1 num_list = [2, 3, 4, 6, 8]
2 print(num_list[-2:]) # Prints [6, 8], second last to last elements
3 print(num_list[:2]) # Prints [2, 3], first 2 elements
4 print(num_list[1:4]) # Prints [3, 4, 6], elements from index 1 to 4
5 print(num_list[1:-2]) # Prints [3, 4], elements from index 1 to second last
```

Membership

```
1 num_list = [2, 3, 4, 6, 8]
2 test_1 = 3 in num_list
3 print(test_1) # True
4 test_2 = 4 not in num_list
5 print(test_2) # False
```

Tuples

```
1 tp_1 = (1, 2, 3)
2 tp_2 = 4, 5, 6
3 print(tp_1) # Prints (1, 2, 3)
4 print(tp_2) # Prints (4, 5, 6)
5 tp_1[1] = 2 # Error
```

Sets

```
1 items = set()
2 items.add("rice")
3 items.add("burger")
4 items.add(5)
5 items.add("mango")
6 items.remove("mango")
7 print(items)
```

Dictionaries

```
1 scores = {}
2 scores = {"Madeline": 98,
3           "Melody": 99,
4           "Jane": 88}
5
6 print(scores["Madeline"]) # Prints 98
7 print(scores["Leo"]) # Error
8
9 scores["Luke"] = 87
10 print(scores["Luke"]) # Prints 87
11
12 have_Melody = "Melody" in scores # Prints True
13 print(have_Melody)
14
15 have_Joe = "Joe" in scores # Prints False
16 print(have_Joe)
```

Decision Making

```
1 expensive = True
2 good = False
3
4 if expensive:
5     print("It is expensive")
6
7 if not expensive:
8     print("It is cheap")
9
10 if expensive and good:
11     print("It is not a deal")
12
13 if expensive or good:
14     print("It might not be a deal")
```

Comparison

```
1 room_temp = 70
2 temp_now = 50
3
4 if temp_now < room_temp:
5     print("cold")
6 elif temp_now > room_temp:
7     print("hot")
8 else:
9     print("comfortable")
10
11 if temp_now == 70:
12     print("room temperature")
13
14 if temp_now != 70:
15     print("not room temperature")
```

Ternary Operator

```
1 number = 10
2 parity = "even" if number%2==0 else "odd"
3 print(parity)
```

For Loop

```
1 for i in range(1, 10, 2): # step is 2
2     print(i) # 1, 3, 5, 7, 9
3
4 for i in range(5): # default step is 1
5     print(i) # 0, 1, 2, 3, 4
6
7 num_list = [1, 3, 4]
8 for i in num_list: # loop through each element
9     print(i) # 1, 3, 4
```

Break

```
1 for k in range(5):
2     print(k) # print up to 3
3     if (k == 3):
4         break
```

Continue

```
1 for num in range(7):
2     if (num == 4):
3         continue
4     print(num) # print 0 through 7, except 4
```

Enumerating

```
1 class_list = ["math", "science", "engineering"]
2 # Simplify looping with counters
3 for count, my_class in enumerate(class_list):
4     print(count, my_class) # prints 0 math 1 science 2 engineering
5
6 for count, my_class in enumerate(class_list, start=1):
7     print(count, my_class) # prints 1 math 2 science 3 engineering
8
9 str1 = "python"
10 # changing start index to 2 from 0
11 print(list(enumerate(str1, 2)))
```

List Comprehension

```
1 numbers = [1, 2, 3]
2 squares = [n**2 for n in numbers]
3 print(squares) # prints [1, 4, 9]
4
5 even_squares = [n**2 for n in numbers if n%2 == 0]
6 print(even_squares) # prints 4
7
8 class_list = ["math", "science", "engineering"]
9 newlist = [x for x in class_list if "e" in x]
10 print(newlist) # prints ['science', 'engineering']
```

While Loop

```
1 counter = 0
2 target = 9
3
4 while counter <= target: # Repeat until the condition is false
5     print(counter)
6     counter += 1 # Increase counter by 1
```

Map, Filter, Reduce

```
1 my_list = list([1,2,3,4,5,6,7])
2
3 squares = map(lambda x: x**2, my_list)
4 print(list(squares))
5
6 # Use anonymous function to filter and compare
7 even_list = filter(lambda x: x%2 == 0, my_list)
8 print(list(even_list)) # Printing the filter
9
10 from functools import reduce
11
12 total = reduce(lambda s, x: s+x, my_list)
13 print(total)
14
15 max = reduce(lambda a,b: a if a>b else b, my_list)
16 print(max)
```

Exceptions Handling

```
1 try:
2     f = open("test.txt", 'r')
3 except FileNotFoundError as fe:
4     print(fe)
5     print('Creating file...')
6     f = open("test.txt", 'w')
7     f.write('2022')
8 else:
9     data=f.readline(1)
10    print(data)
11 finally:
12    print('File closed')
13    f.close()
```

Raising Exceptions

```
1 x = 3
2
3 if not type(x) is str:
4     raise TypeError("String only")
5
6 if len(x) < 3:
7     raise Exception("Too short")
```

