Python 3 Cheat Sheet

MLtrends

Comments

```
# This is a one line comment

'''
This is a multi-line
comment
'''

"""
This is a multi-line
comment too
"""
```

String

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

String Concatenation

```
str_1 = "Hello"
str_2 = "World"

sentence = str_1 + ", " + str_2 + "!"
# Hello, World!

words = ["This", "is", "an", "amazing", "thing"]
print(' '.join(words))
# This is an amazing thing
print(', '.join(words))
# This, is, an, amazing, thing
```

Input/Output

```
str_1 = "Python 3"
str_2 = "Cheat sheet"

print("%s %s" %(str_1,str_2)) # Python 3 Cheat sheet
print("{} {}".format(str_1, str_2)) # Python 3 Cheat sheet
print("{0} {1}".format(str_1, str_2)) # Python 3 Cheat sheet
print("{1} {0}".format(str_1, str_2)) # Cheat sheet Python 3
print(f"{str_1} {str_2}") # Python 3 Cheat sheet

your_age = input("How old are you?") # The input is string type
print(f"{60-int(your_age)} years to retire") # Convert string to integer
```

Defining and Calling a Function

```
def display_something(): # Define a function
    print("This is a demo!")

display_something() # Call the function
```

Function with Arguments

```
def my_sum1(x, y):
    res = x + y
    return res

my_sum1(3, 4) # Returns 7
my_sum1(y = 4, x = 3) # Return 7

# Add optional argument for the function
def my_sum2(x, opt = 0):
    res = 2*x + opt
    return res

my_sum2(3), my_sum2(3, opt = 4)
```

Function with Variable No. of Arguments

```
# *args for non-key worded, variable number of arguments
def my_args_fun(*args):
    print(args)

my_args_fun("non", "key", "worded", "variable no. of arguments")

# *kwargs for variable number of keyword arguments
def my_kwargs_fun(**kwargs):
    print(kwargs)

my_kwargs_fun(first =32, mid =1, last=22)

# Using *args and **kwargs for a function
def my_fun(*args,**kwargs):
    print("args: ", args)
    print("kwargs: ", kwargs)
```

Lambda Function

```
import math

def my_sqrt(x): # define a regular function
    return math.sqrt(x)
print(my_sqrt(4))

g = lambda x: math.sqrt(x) # define a online function
print(g(4))
```

Global Variable

```
counter = 0
def my_count():
    global counter
    counter = counter + 1
    print(counter)
my_count()
```

Lists

```
str_list = ["hello", "world"]
mixed_list = [2, 3.1, "hi"]
list_of_list = [str_list, mixed_list]

print(str_list[0])
print(len(str_list))
print(len(mixed_list))
print(len(list_of_list))
```

List Methods

```
int_list = [3, 2, 5]
int_list.append(7) # Append a number to the list
print(int_list) # [3, 2, 5, 7]

int_list.extend([1, 6]) # Extend numbers to the list
print(int_list) # [3, 2, 5, 7, 1, 6]

print(int_list + [1, 6]) # Extend numbers to the list
# [3, 2, 5, 7, 1, 6, 1, 6]

int_list.pop() # Remove the last elememt from the list
print(int_list) # [3, 2, 5, 7, 1]

int_list.pop(3) # Remove the last elememt at index 3
print(int_list) # [3, 2, 5, 1]

int_list.remove(3) # Remove the first matching element
print(int_list) # [2, 5, 1]

int_list.insert(2, 8) # Insert 5 into index 2
print(int_list) # [2, 5, 8, 1]
```

List Unpacking

```
nums = [18, 88]
age, math = nums
print("The math is %d" %math) # The math is 88
age, _ = nums # Only interested in age
print(age) # 18

r = range(5) # Create a range from 0 to 4
print(*r) # Prints [0 1 2 3 4], unpack the range

print(r[0]) # Prints 0 , number at 0 index
print(r[-1]) # Prints 4, number at the last index
print(r[-2]) # Prints 3, number at the second last index
```

Slicing

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

Membership

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

Tuples

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

Sets

```
items = set()
items.add("rice")
items.add("burger")
items.add(5)
items.add("mango")
items.remove("mango")
print(items)
```

Dictionaries

```
scores = {}
scores = {"Madeline": 98,
          "Melody": 99,
          "Jane": 88}

print(scores["Madeline"]) # Prints 98
print(scores["Leo"]) # Error

scores["Luke"] = 87
print(scores["Luke"]) # Prints 87

have_Melody = "Melody" in scores # Prints True
print(have_Melody)

have_Joe = "Joe" in scores # Prints False
print(have_Joe)
```

Decision Making

```
expensive = True
good  = False

if expensive:
    print("It is expensive")

if not expensive:
    print("It is cheap")

if expensive and good:
    print("It is not a deal")

if expensive or good:
    print("It might not be a deal")
```

Comparison

```
room_temp = 70
temp_now = 50

if temp_now < room_temp:
    print("cold")
elif temp_now > room_temp:
    print("hot")
else:
    print("confortable")

if temp_now == 70:
    print("room temperature")

if temp_now != 70:
    print("not room temperature")
```

Ternary Operator

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

For Loop

```
for i in range(1, 10, 2): # step is 2
    print(i) # 1, 3, 5, 7, 9

for i in range(5): # default step is 1
    print(i) # 0, 1, 2, 3, 4

num_list = [1, 3, 4]
for i in num_list: # loop through each element
    print(i) # 1, 3, 4
```

Break

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

Continue

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

Enumerating

```
class_list = ["math", "science", "engineering"]
# Simplify looping with counters
for count, my_class in enumerate(class_list):
    print(count, my_class) # prints 0 math 1 science 2 engineering

for count, my_class in enumerate(class_list, start=1):
    print(count, my_class) # prints 1 math 2 science 3 engineering

str1 = "python"
# changing start index to 2 from 0
print (list(enumerate(str1, 2)))
```

List Comprehension

```
numbers = [1, 2, 3]
squares = [n**2 for n in numbers]
print(squares) # prints [1, 4, 9]

even_squares = [n**2 for n in numbers if n%2 == 0]
print(even_squares) # prints 4

class_list = ["math", "science", "engineering"]
newlist = [x for x in class_list if "e" in x]
print(newlist) # prints ['science', 'engineering']
```

While Loop

```
counter = 0
target = 9

while counter <= target: # Repeat until the condition is false
    print(counter)
    counter += 1 # Increase counter by 1
```

Map, Filter, Reduce

```
my_list = list([1,2,3,4,5,6,7])

squares = map(lambda x: x**2, my_list)
print([*squares])

# Use anonymous function to filter and compare
even_list = filter(lambda x: x%2 == 0, my_list)
print([*even_list]) # Printing the filter

from functools import reduce

total = reduce(lambda s, x: s+x, my_list)
print(total)

max = reduce(lambda a,b: a if a>b else b, my_list)
print(max)
```

Classes

```
class MyClass:
    # Automatically invokes __init__()
    # for the newly-created class instance
    def __init__(self, name):
        self.name = name # Attributes
        self.skills = []

    def add_skills(self, skill): # Method
        self.skills.append(skill)

p1 = MyClass("Jeo") # Create an instance
p1.add_skills("Python")
p1.add_skills("C")

p2 = MyClass("John") # Create an instance
p2.add_skills("football")
p2.add_skills("driving")

p1.skills # ['Python', 'C']
p2.skills # ['football', 'driving']
```

Exceptions Handling

```
try:
    f = open("test.txt", 'r')
except FileNotFoundError as fe:
    print(fe)
    print ('Creating file...')
    f = open("test.txt", 'w')
    f.write('2022')
else:
    data=f.readline(1)
    print(data)
finally:
    print ('File closed')
    f.close()
```

Raising Exceptions

```
x = 3

if not type(x) is str:
    raise TypeError("String only")

if len(x) < 3:
  raise Exception("Too short")
```