Python3 Cheatsheet

Strings

```
# Multiline strings
multiline_string = """This is a
multiline string"""

# Length of a string
length = len("Hello")  # Output: 5

# Checking if a string starts with a specific substring
starts_with = "Hello, World".startswith("Hello")  # Output: True

# Checking if a string ends with a specific substring
ends_with = "Hello, World".endswith("World")  # Output: True

# String interpolation
name = "Alice"
age = 30
message = f"My name is {name} and I am {age} years old."

# String formatting
formatted_message = "My name is {} and I am {} years old.".format(name,
age)

# Replacing substrings in a string
new_string = "Hello, World".replace("World", "Python")  # Output: Hello,
Python

# Splitting a string into a list of substrings
words = "Hello, World".split(",")  # Output: ['Hello', ' World']

# Splicing a string
spliced_string = "Hello, World"[1:5]  # Output: "ello"

# Converting a string to lowercase
lowercase_string = "Hello".lower()  # Output: "hello"
```

```
# Converting a string to title case
title_case_string = "hello world".title()  # Output: "Hello World"

# Splitting a string into lines
lines = "Hello\nWorld".splitlines()  # Output: ['Hello', 'World']
```

Lists

```
# Creating a list
my_list = list(range(1, 6))  # Output: [1, 2, 3, 4, 5]

# Slicing a list
sliced_list = my_list[1:4]  # Output: [2, 3, 4]

# Length of a list
length = len(my_list)  # Output: 5

# Sum of all elements in a list
total = sum(my_list)  # Output: 15

# Minimum value in a list
minimum = min(my_list)  # Output: 1

# Maximum value in a list
maximum = max(my_list)  # Output: 5

# Appending an element to a list
my_list.append(6)  # Output: [1, 2, 3, 4, 5, 6]

# Sorting a list
my_list.sort()  # Output: [1, 2, 3, 4, 5, 6]

# Joining list elements into a string
joined_string = ", ".join(map(str, my_list))  # Output: "1, 2, 3, 4, 5, 6"

# List comprehensions
squared = [x**2 for x in my_list]  # Output: [1, 4, 9, 16, 25, 36]

# Removing an element from a list by value
my_list.remove(3)  # Output: [1, 2, 4, 5, 6]

# Reversing a list
reversed_list = my_list[::-1]  # Output: [6, 5, 4, 2, 1]
```

Dictionaries

```
# Creating a dictionary
my_dict = {'a': 1, 'b': 2, 'c': 3}
```

```
# Accessing values using keys
value = my_dict['a']  # Output: 1

# Getting all keys
keys = my_dict.keys()  # Output: dict_keys(['a', 'b', 'c'])

# Getting all values
values = my_dict.values()  # Output: dict_values([1, 2, 3])

# Getting key-value pairs as tuples
items = my_dict.items()  # Output: dict_items([('a', 1), ('b', 2), ('c',
3)])

# Enumerating through dictionary items
for index, (key, value) in enumerate(my_dict.items()):
    print(index, key, value)
```

Sets

```
# Creating a set
my_set = {1, 2, 3}

# Set operations (union, intersection, difference)
union_set = {1, 2, 3} | {3, 4, 5}  # Output: {1, 2, 3, 4, 5}
intersection_set = {1, 2, 3} & {3, 4, 5}  # Output: {3}
difference_set = {1, 2, 3} - {3, 4, 5}  # Output: {1, 2}
```

Loops and Control Flow

```
# If statement
x = 10
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is less than 5")

# While loop
x = 0
while x < 5:
    print(x)
    x += 1
    if x == 3:
        continue  # Skip the rest of the loop body and continue with the
next iteration
    if x == 4:
```

```python
        break  # Exit the loop completely

# For loop
for i in range(5):
    print(i)
```

Modules

```python
# Importing a module
import math

# Importing specific names from a module
from math import pi
```

Files

```python
# Opening a file
file = open("filename.txt", "r")

# Reading from a file
content = file.read()

# Writing to a file
with open("filename.txt", "w") as file:
    file.write("Hello, world!")
```

Exceptions

```python
# Exception handling
try:
    result = 10 / 0
except ZeroDivisionError as e:
    print("Error:", e)
```

Functions

```python
# arguments
def add(a, b):
    return a + b

result = add(3, 4)  # Output: 7

# default values (default values in function parameters)
```

```
def greet(name="World"):
    print("Hello, " + name)

greet()  # Output: Hello, World
greet("Alice")  # Output: Hello, Alice

# keyword arguments
def greet_with_age(name, age):
    print(f"Hello, {name}! You are {age} years old.")

greet_with_age(name="Alice", age=30)  # Output: Hello, Alice! You are 30
years old.

# docstrings (docstrings for documenting functions)
def multiply(a, b):
    """
    This function multiplies two numbers.
    """
    return a * b

# *args (arbitrary number of arguments)
def add(*args):
    total = 0
    for num in args:
        total += num
    return total

result = add(1, 2, 3, 4)  # Output: 10

# **kwargs (arbitrary number of keyword arguments)
def greet_with_details(**kwargs):
    if 'name' in kwargs and 'age' in kwargs:
        print(f"Hello, {kwargs['name']}! You are {kwargs['age']} years
old.")
    else:
        print("Hello, Unknown!")

greet_with_details(name="Alice", age=30)  # Output: Hello, Alice! You are
30 years old.


# Taking command line arguments
import sys

def main():
    # Taking command line arguments
    if len(sys.argv) == 3:
        arg1 = int(sys.argv[1])
        arg2 = int(sys.argv[2])
        print(f"Result: {arg1 + arg2}")
    else:
        print("Usage: python script.py <arg1> <arg2>")
```

```python
if __name__ == "__main__":
    main()
```

OOP (Object-Oriented Programming)

```python
# Class initialization
class MyClass:
    def __init__(self, x):
        self.x = x

    def print_x(self):
        print(self.x)

# Defining a method inside a class
class MyClass:
    def my_method(self):
        pass

# Using class methods
class MyClass:
    @classmethod
    def my_class_method(cls):
        pass

# Inheritance
class ChildClass(ParentClass):
    pass
```