Python Cheat Sheet

Python Basics:

Descriptive Variable Names:

```
# Bad Example
x = 10
y = 20
result = x + y

# Good Example
num1 = 10
num2 = 20
sum_of_numbers = num1 + num2
```

Follow PEP 8 Style Guidelines:

```
# Use snake_case for variable names
my_variable = 42

# Use descriptive names
user_age = 25
```

Use Built-in Functions:

```
# Example: Built-in function
numbers = [1, 2, 3, 4, 5]
print(sum(numbers))  # Output: 15
```

List Comprehensions:

```
# Example: List comprehension
numbers = [1, 2, 3, 4, 5]
squared_numbers = [x ** 2 for x in numbers]
print(squared_numbers)  # Output: [1, 4, 9, 16, 25]
```

Data Structures:

Sets for Membership Tests and Removing Duplicates:

```
# Example: Sets
unique_numbers = {1, 2, 3, 4, 5}
if 3 in unique_numbers:
    print("3 is in the set")

# Removing duplicates
numbers = [1, 2, 3, 4, 2, 3, 5]
unique_numbers = set(numbers)
print(unique_numbers)  # Output: {1, 2, 3, 4, 5}
```

Dictionaries for Key-Value Mappings:

```
# Example: Dictionary
student = {
    "name": "Alice",
    "age": 25,
    "major": "Computer Science"
}
print(student["name"])  # Output: Alice
```

Functions and Lambdas:

Modular and Reusable Functions:

```
# Example: Function
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))  # Output: Hello, Alice!
```

Default and Keyword Arguments:

```
# Example: Function with default argument
def greet(name="World"):
    return f"Hello, {name}!"

print(greet())  # Output: Hello, World!
print(greet("Alice"))  # Output: Hello, Alice!
```

Lambda Functions:

```
# Example: Lambda function
add = lambda x, y: x + y
print(add(3, 4))  # Output: 7
```

Decorators:

```
# Example: Decorator
def my_decorator(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

@my_decorator
def say_hello():
    print("Hello!")

say_hello()
```

Generators:

```
# Example: Generator
def countdown(n):
    while n > 0:
        yield n
        n -= 1
```

```
for i in countdown(5):
    print(i)  # Output: 5, 4, 3, 2, 1
```

Control Flow:

List Comprehensions and Generator Expressions:

```
# Example: List comprehension
numbers = [1, 2, 3, 4, 5]
squared_numbers = [x ** 2 for x in numbers]
print(squared_numbers)  # Output: [1, 4, 9, 16, 25]
```

Ternary Operator:

```
# Example: Ternary operator
x = 10
y = 20
result = x if x > y else y
print(result)  # Output: 20
```

Context Managers:

```
# Example: Context manager
with open("file.txt", "r") as file:
    contents = file.read()
```

Itertools Module:

```
# Example: itertools module
import itertools

# Cartesian product
for pair in itertools.product([1, 2], ['a', 'b']):
    print(pair)
```

Exception Handling:

~~~
# Example: Exception handling
try:
   Certainly! Here's the HTML version of the content for you to copy:

```html
~~~

Python Cheat Sheet

Python Basics:

Descriptive Variable Names:

```
# Bad Example
x = 10
y = 20
result = x + y

# Good Example
num1 = 10
num2 = 20
sum_of_numbers = num1 + num2
```

Follow PEP 8 Style Guidelines:

```
# Use snake_case for variable names
my_variable = 42

# Use descriptive names
user_age = 25
```

Use Built-in Functions:

```
# Example: Built-in function
numbers = [1, 2, 3, 4, 5]
print(sum(numbers))  # Output: 15
```

List Comprehensions:

```python
# Example: List comprehension
numbers = [1, 2, 3, 4, 5]
squared_numbers = [x ** 2 for x in numbers]
print(squared_numbers)  # Output: [1, 4, 9, 16, 25]
```

Data Structures:

Sets for Membership Tests and Removing Duplicates:

```python
# Example: Sets
unique_numbers = {1, 2, 3, 4, 5}
if 3 in unique_numbers:
    print("3 is in the set")

# Removing duplicates
numbers = [1, 2, 3, 4, 2, 3, 5]
unique_numbers = set(numbers)
print(unique_numbers)  # Output: {1, 2, 3, 4, 5}
```

Dictionaries for Key-Value Mappings:

```python
# Example: Dictionary
student = {
    "name": "Alice",
    "age": 25,
    "major": "Computer Science"
}
print(student["name"])  # Output: Alice
```

Functions and Lambdas:

Modular and Reusable Functions:

```python
# Example: Function
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))  # Output: Hello, Alice!
```

Default and Keyword Arguments:

```python
# Example: Function with default argument
def greet(name="World"):
    return f"Hello, {name}!"

print(greet())  # Output: Hello, World!
print(greet("Alice"))  # Output: Hello, Alice!
```

Lambda Functions:

```python
# Example: Lambda function
add = lambda x, y: x + y
print(add(3, 4))  # Output: 7
```

Decorators:

```python
# Example: Decorator
def my_decorator(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

@my_decorator
def say_hello():
    print("Hello!")

say_hello()
```

Generators:

```
# Example: Generator
def countdown(n):
    while n > 0:
        yield n
        n -= 1

for i in countdown(5):
    print(i)  # Output: 5, 4, 3, 2, 1
```

Control Flow:

List Comprehensions and Generator Expressions:

```
# Example: List comprehension
numbers = [1, 2, 3, 4, 5]
squared_numbers = [x ** 2 for x in numbers]
print(squared_numbers)  # Output: [1, 4, 9, 16, 25]
```

Ternary Operator:

```
# Example: Ternary operator
x = 10
y = 20
result = x if x > y else y
print(result)  # Output: 20
```

Context Managers:

```
# Example: Context manager
with open("file.txt", "r") as file:
    contents = file.read()
```

Itertools Module:

```
# Example: itertools module
import itertools

# Cartesian product
for pair in itertools.product([1, 2], ['a', 'b']):
    print(pair)
```

Exception Handling:

```
# Example: Exception handling
try:


 result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

Object-Oriented Programming (OOP):

Classes and Objects:

```
# Example: Class
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self):
        return "Woof!"

my_dog = Dog("Buddy")
print(my_dog.bark())  # Output: Woof!
```

Inheritance and Composition:

```
# Example: Inheritance
class Animal:
    def speak(self):
        raise NotImplementedError("Subclass must implement abstract
method")

class Dog(Animal):
    def speak(self):
        return "Woof!"

class Cat(Animal):
    def speak(self):
        return "Meow!"

my_dog = Dog()
print(my_dog.speak())  # Output: Woof!
```

Dunder Methods:

```
# Example: Dunder method
class Dog:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return f"Dog: {self.name}"

my_dog = Dog("Buddy")
print(my_dog)  # Output: Dog: Buddy
```

Class vs Instance Attributes:

```
# Example: Class vs Instance attributes
class Car:
    # Class attribute
    wheels = 4

    def __init__(self, make, model):
        # Instance attributes
        self.make = make
        self.model = model

my_car = Car("Toyota", "Camry")
print(my_car.wheels)  # Output: 4
```

Debugging and Testing:

Print Statements:

```
# Example: Print statements for debugging
x = 10
y = 0
print(x / y)  # Output: ZeroDivisionError
```

Unit Tests with Unittest:

```
# Example: Unit test with unittest
import unittest

def add(x, y):
    return x + y

class TestAdd(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(3, 4), 7)

if __name__ == "__main__":
    unittest.main()
```

Performance Optimization:

Profile Your Code:

```
# Example: Profile your code
import cProfile

def test_function():
    # Your code here
    pass

cProfile.run('test_function()')
```

Use Efficient Data Structures and Algorithms:

~~~
# Example: Efficient data structure
from collections import defaultdict

# Use defaultdict instead of manual checking for missing keys
my_dict = defaultdict(list)




```
~~~