Complete Python Cheatsheet

Learn Everything AI - Shivam Modi

June 2, 2023

Contents

1 Introduction

Python is a high-level, interpreted programming language known for its simplicity and readability. It has a large user community and a wide range of libraries that make it versatile and powerful. This cheatsheet provides a quick reference for Python programming concepts.

2 Variables and Data Types

2.1 Variables

A variable is a named location in memory used to store data. In Python, variable names are case-sensitive and can contain letters, numbers, and underscores (_).

```
# Variable assignment
x = 5
y = "Hello, World!"
```

2.2 Data Types

Python has several built-in data types, including:

- **Numeric**: int, float, complex
- **Sequence**: list, tuple, range
- **Mapping**: dict
- **Set**: set, frozenset
- **Boolean**: bool
- **None**: NoneType

3 Control Flow

3.1 Conditional Statements

Conditional statements allow you to perform different actions based on conditions.

```
# If statement
if condition:
    statement(s)

# If-else statement
if condition:
    statement(s)
else:
    statement(s)

# If-elif-else statement
if condition1:
    statement(s)
elif condition2:
    statement(s)
else:
    statement(s)
```

3.2 Loops

Loops enable repeated execution of a block of code.

```
# For loop
for item in sequence:
    statement(s)

# While loop
while condition:
    statement(s)

# Loop control statements
break
\begin{lstlisting}[language=Python]
# For loop (continued)
for item in sequence:
    if condition:
        break
    statement(s)

# Continue statement
for item in sequence:
    if condition:
        continue
    statement(s)

# Range function
for i in range(start, stop, step):
    statement(s)

# Nested loops
for item in sequence:
    for element in nested_sequence:
        statement(s)

# While loop (continued)
while condition:
    if condition:
        break
    statement(s)

# Continue statement
while condition:
    if condition:
        continue
    statement(s)
```

4 Functions

Functions are reusable blocks of code that perform a specific task.

```
# Defining a function
def function_name(parameters):
    statement(s)
    return value

# Function call
result = function_name(arguments)
```

```
# Default parameters
def function_name(parameter=value):
    statement(s)

# Variable number of arguments
def function_name(*args):
    statement(s)

# Keyword arguments
def function_name(**kwargs):
    statement(s)

# Lambda functions
lambda arguments: expression
```

5 Lists

A list is a collection of items that are ordered and mutable.

```
# Creating a list
my_list = [item1, item2, item3]

# Accessing list elements
item = my_list[index]

# Slicing a list
new_list = my_list[start:end]

# Modifying list elements
my_list[index] = new_value

# Adding elements to a list
my_list.append(item)

# Removing elements from a list
my_list.remove(item)

# List operations
combined_list = list1 + list2
repeated_list = my_list * n
```

6 Dictionaries

A dictionary is an unordered collection of key-value pairs.

```
# Creating a dictionary
my_dict = {"key1": value1, "key2": value2}

# Accessing dictionary values
value = my_dict["key"]

# Modifying dictionary values
my_dict["key"] = new_value

# Adding key-value pairs to a dictionary
my_dict["new_key"] = value
```

```
# Removing key-value pairs from a dictionary
del my_dict["key"]

# Dictionary operations
keys = my_dict.keys()
values = my_dict.values()
items = my_dict.items()
```

7 Modules and Packages

Modules are Python files that contain reusable code, and packages are directories that contain multiple modules.

```
# Importing a module
import module_name

# Importing specific items from a module
from module_name import item

# Importing an entire module with an alias
import module_name as alias

# Importing specific items from a module with aliases
from module_name import item as alias

# Importing all items from a module
from module_name import *

# Creating a package
__init__.py

# Importing a module from a package
from package_name import module_name
```

8 File Handling

Python provides functions for reading from and writing to files.

```
# Opening a file
file = open(filename, mode)

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

# Writing to a file
file.write(content)

# Appending to a file
file = open(filename, "a")
file.write(content)

# Closing a file
file.close()
```

9 Exception Handling

Exception handling allows you to handle and manage errors that occur during the execution of your program.

```
# Try-except block
try:
    statement(s)
except ExceptionType:
    statement(s)

# Handling multiple exceptions
try:
    statement(s)
except ExceptionType1:
    statement(s)
except ExceptionType2:
    statement(s)

# Finally block
try:
    statement(s)
except ExceptionType:
    statement(s)
finally:
    statement(s)

# Raising an exception
raise ExceptionType("Error message")
```

10 Classes and Objects

Classes are blueprints for creating objects, and objects are instances of a class.

```
# Defining a class
class ClassName:
    def __init__(self, parameters):
        self.attribute = value

    def method(self, parameters):
        statement(s)

# Creating an object
object_name = ClassName(arguments)

# Accessing attributes
value = object_name.attribute

# Calling methods
object_name.method(arguments)
```

11 Inheritance and Polymorphism

Inheritance allows you to create a new class that inherits the properties and methods of an existing class, while polymorphism enables objects of different classes to be used interchangeably.

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

```
    def __init__(self, parameters):
        super().__init__(parameters)
        self.attribute = value

    def method(self, parameters):
        statement(s)

# Polymorphism
class Class1:
    def method(self, parameters):
        statement(s)

class Class2:
    def method(self, parameters):
        statement(s)

def function_name(object):
    object.method(arguments)
```

12 Error and Exception Handling

Python provides built-in error and exception handling mechanisms to catch and handle errors during program execution.

```
# Try-except block
try:
    statement(s)
except ExceptionType as e:
    statement(s)
    print(e)

# Handling multiple exceptions
try:
    statement(s)
except ExceptionType1:
    statement(s)
except ExceptionType2:
    statement(s)

# Finally block
try:
    statement(s)
except ExceptionType:
    statement(s)
finally:
    statement(s)

# Raising an exception
raise ExceptionType("Error message")
```

13 Conclusion

This Python cheatsheet covers the fundamental concepts and syntax of Python programming. It serves as a quick reference guide for beginners. Keep exploring and practicing to enhance your Python skills!