

Complete Python Cheatsheet

Learn Everything AI - Shivam Modi

June 2, 2023

Contents

1	Introduction	2
2	Variables and Data Types	2
2.1	Variables	2
2.2	Data Types	2
3	Control Flow	2
3.1	Conditional Statements	2
3.2	Loops	3
4	Functions	3
5	Lists	4
6	Dictionaries	4
7	Modules and Packages	5
8	File Handling	5
9	Exception Handling	6
10	Classes and Objects	6
11	Inheritance and Polymorphism	6
12	Error and Exception Handling	7
13	Conclusion	7

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 (_).

```
1 # Variable assignment
2 x = 5
3 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.

```
1 # If statement
2 if condition:
3     statement(s)
4
5 # If-else statement
6 if condition:
7     statement(s)
8 else:
9     statement(s)
10
11 # If-elif-else statement
12 if condition1:
13     statement(s)
14 elif condition2:
15     statement(s)
16 else:
17     statement(s)
```

3.2 Loops

Loops enable repeated execution of a block of code.

```
1 # For loop
2 for item in sequence:
3     statement(s)
4
5 # While loop
6 while condition:
7     statement(s)
8
9 # Loop control statements
10 break
11 \begin{lstlisting}[language=Python]
12 # For loop (continued)
13 for item in sequence:
14     if condition:
15         break
16     statement(s)
17
18 # Continue statement
19 for item in sequence:
20     if condition:
21         continue
22     statement(s)
23
24 # Range function
25 for i in range(start, stop, step):
26     statement(s)
27
28 # Nested loops
29 for item in sequence:
30     for element in nested_sequence:
31         statement(s)
32
33 # While loop (continued)
34 while condition:
35     if condition:
36         break
37     statement(s)
38
39 # Continue statement
40 while condition:
41     if condition:
42         continue
43     statement(s)
```

4 Functions

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

```
1 # Defining a function
2 def function_name(parameters):
3     statement(s)
4     return value
5
6 # Function call
7 result = function_name(arguments)
```

```

8
9 # Default parameters
10 def function_name(parameter=value):
11     statement(s)
12
13 # Variable number of arguments
14 def function_name(*args):
15     statement(s)
16
17 # Keyword arguments
18 def function_name(**kwargs):
19     statement(s)
20
21 # Lambda functions
22 lambda arguments: expression

```

5 Lists

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

```

1 # Creating a list
2 my_list = [item1, item2, item3]
3
4 # Accessing list elements
5 item = my_list[index]
6
7 # Slicing a list
8 new_list = my_list[start:end]
9
10 # Modifying list elements
11 my_list[index] = new_value
12
13 # Adding elements to a list
14 my_list.append(item)
15
16 # Removing elements from a list
17 my_list.remove(item)
18
19 # List operations
20 combined_list = list1 + list2
21 repeated_list = my_list * n

```

6 Dictionaries

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

```

1 # Creating a dictionary
2 my_dict = {"key1": value1, "key2": value2}
3
4 # Accessing dictionary values
5 value = my_dict["key"]
6
7 # Modifying dictionary values
8 my_dict["key"] = new_value
9
10 # Adding key-value pairs to a dictionary
11 my_dict["new_key"] = value

```

```

12
13 # Removing key-value pairs from a dictionary
14 del my_dict["key"]
15
16 # Dictionary operations
17 keys = my_dict.keys()
18 values = my_dict.values()
19 items = my_dict.items()

```

7 Modules and Packages

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

```

1 # Importing a module
2 import module_name
3
4 # Importing specific items from a module
5 from module_name import item
6
7 # Importing an entire module with an alias
8 import module_name as alias
9
10 # Importing specific items from a module with aliases
11 from module_name import item as alias
12
13 # Importing all items from a module
14 from module_name import *
15
16 # Creating a package
17 __init__.py
18
19 # Importing a module from a package
20 from package_name import module_name

```

8 File Handling

Python provides functions for reading from and writing to files.

```

1 # Opening a file
2 file = open(filename, mode)
3
4 # Reading from a file
5 content = file.read()
6 lines = file.readlines()
7
8 # Writing to a file
9 file.write(content)
10
11 # Appending to a file
12 file = open(filename, "a")
13 file.write(content)
14
15 # Closing a file
16 file.close()

```

9 Exception Handling

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

```
1 # Try-except block
2 try:
3     statement(s)
4 except ExceptionType:
5     statement(s)
6
7 # Handling multiple exceptions
8 try:
9     statement(s)
10 except ExceptionType1:
11     statement(s)
12 except ExceptionType2:
13     statement(s)
14
15 # Finally block
16 try:
17     statement(s)
18 except ExceptionType:
19     statement(s)
20 finally:
21     statement(s)
22
23 # Raising an exception
24 raise ExceptionType("Error message")
```

10 Classes and Objects

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

```
1 # Defining a class
2 class ClassName:
3     def __init__(self, parameters):
4         self.attribute = value
5
6     def method(self, parameters):
7         statement(s)
8
9 # Creating an object
10 object_name = ClassName(arguments)
11
12 # Accessing attributes
13 value = object_name.attribute
14
15 # Calling methods
16 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.

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

```

3     def __init__(self, parameters):
4         super().__init__(parameters)
5         self.attribute = value
6
7     def method(self, parameters):
8         statement(s)
9
10 # Polymorphism
11 class Class1:
12     def method(self, parameters):
13         statement(s)
14
15 class Class2:
16     def method(self, parameters):
17         statement(s)
18
19 def function_name(object):
20     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.

```

1 # Try-except block
2 try:
3     statement(s)
4 except ExceptionType as e:
5     statement(s)
6     print(e)
7
8 # Handling multiple exceptions
9 try:
10     statement(s)
11 except ExceptionType1:
12     statement(s)
13 except ExceptionType2:
14     statement(s)
15
16 # Finally block
17 try:
18     statement(s)
19 except ExceptionType:
20     statement(s)
21 finally:
22     statement(s)
23
24 # Raising an exception
25 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!