Cheat Sheet for comprehensive Python

Basic Syntax and Structure

- Comments

- Single-line: `# This is a comment`
- Multi-line: `""" This is a multi-line comment """`

- Variables and Data Types

- `x = 10` (integer)
- `y = 3.14` (float)
- `name = "Alice"` (string)
- `is_valid = True` (boolean)

- Basic Operations

- Addition: `x + y`
- Subtraction: `x - y`
- Multiplication: `x * y`
- Division: `x / y`
- Floor Division: `x // y`
- Modulus: `x % y`

- Exponentiation: `x ** y`

- String Operations

- Concatenation: `"Hello" + "World"`
- Repetition: `"Hi" * 3`
- Slicing: `name[0:3]`
- Length: `len(name)`

Control Flow

- Conditional Statements

- `if condition:`
- `elif condition:`
- `else:`

- Loops

- `for item in iterable:`
- `while condition:`

- **Loop Control Statements**

- `break`: Exit the loop
- `continue`: Skip the current iteration
- `pass`: Do nothing (placeholder)

Functions

- **Defining Functions**

- `def function_name(parameters):`
- `return value`

- **Lambda Functions**

- `lambda arguments: expression`
- Example: `add = lambda x, y: x + y`

- **Function Arguments**

- Positional: `def func(a, b):`
- Keyword: `func(a=1, b=2)`
- Default: `def func(a=1, b=2):`

- Variable-length: `def func(*args, **kwargs):`

Data Structures

- **Lists**

- Creation: `my_list = [1, 2, 3]`
- Access: `my_list[0]`
- Append: `my_list.append(4)`
- Extend: `my_list.extend([5, 6])`
- Insert: `my_list.insert(1, 1.5)`
- Remove: `my_list.remove(2)`
- Pop: `my_list.pop()`
- Sort: `my_list.sort()`
- Reverse: `my_list.reverse()`

- **Tuples**

- Creation: `my_tuple = (1, 2, 3)`
- Immutable: Cannot be changed after creation

- **Sets**

- Creation: `my_set = {1, 2, 3}`

- Add: `my_set.add(4)`
- Remove: `my_set.remove(2)`
- Union: `set1 | set2`
- Intersection: `set1 & set2`
- Difference: `set1 - set2`

- Dictionaries

- Creation: `my_dict = {'key1': 'value1', 'key2': 'value2'}`
- Access: `my_dict['key1']`
- Add/Update: `my_dict['key3'] = 'value3'`
- Remove: `del my_dict['key2']`
- Keys: `my_dict.keys()`
- Values: `my_dict.values()`
- Items: `my_dict.items()`

Modules and Packages

- Importing Modules

- `import module_name`
- `from module_name import function_name`
- `import module_name as alias`

- Creating Modules

- Save functions in a `.py` file
- Import the file in another script

- Standard Libraries

- `math`: Mathematical functions
- `os`: Operating system interfaces
- `sys`: System-specific parameters and functions
- `datetime`: Date and time manipulation
- `random`: Generate random numbers

File Handling

- Opening Files

- `file = open('filename.txt', 'r')`
- Modes: `'r'` (read), `'w'` (write), `'a'` (append), `'b'` (binary)

- Reading Files

- `content = file.read()`
- `lines = file.readlines()`

- **Writing to Files**

- `file.write('Hello, World!')`
- `file.writelines(['Line1\n', 'Line2\n'])`

- **Closing Files**

- `file.close()`

- **Using `with` Statement**

- `with open('filename.txt', 'r') as file:`
- Automatically closes the file

Exception Handling

- **Try-Except Block**

- `try:`
- `except ExceptionType as e:`
- `else:` (optional, runs if no exception)
- `finally:` (optional, always runs)

- **Raising Exceptions**

- `raise ExceptionType("Error message")`

Object-Oriented Programming (OOP)

- **Classes and Objects**

- `class MyClass:`
- `def __init__(self, param):`
- `self.attribute = param`

- **Inheritance**

- `class ChildClass(ParentClass):`

- **Encapsulation**

- Public: `self.attribute`
- Protected: `self._attribute`
- Private: `self.__attribute`

- **Polymorphism**

- Method overriding: `def method(self):`

Advanced Features

- **List Comprehensions**

- `[expression for item in iterable if condition]`

- Example: `squares = [x**2 for x in range(10)]`

- **Generators**

- `def my_generator():`
- `yield value`

- Example: `gen = (x**2 for x in range(10))`

- **Decorators**

- `@decorator_function`
- `def my_function():`

- **Context Managers**

- `with context_expression as variable:`
- Example: `with open('file.txt') as f:`

Useful Libraries and Tools

- **NumPy**

- Numerical operations
- `import numpy as np`

- **Pandas**

- Data manipulation and analysis
- `import pandas as pd`

- **Matplotlib**

- Plotting and visualization
- `import matplotlib.pyplot as plt`

- **Requests**

- HTTP requests
- `import requests`

- **Flask/Django**

- Web development frameworks

- `from flask import Flask`

Debugging and Profiling

- **Print Debugging**

- `print(variable)`

- **Using `pdb`**

- `import pdb; pdb.set_trace()`

- **Profiling**

- `import cProfile`
- `cProfile.run('my_function()')`

Tips and Tricks

- **Swapping Variables**

- `a, b = b, a`

- **Unpacking**

- `a, b, *rest = [1, 2, 3, 4]`

- **String Formatting**

- `f"Hello, {name}"`
- `"Hello, {}".format(name)`

- **Reading Input**

- `input("Enter something: ")`

- **Checking Types**

- `isinstance(variable, type)`

- **Documentation Strings**

- `def func():`
- `"""This is a docstring."""`

Conclusion

This cheat sheet covers the essential aspects of Python, from basic syntax to advanced features and useful libraries. Use it as a quick reference to enhance your Python programming skills.

By Ahmed Baheeg Khorshid

ver 1.0