Python Cheatsheet

A comprehensive cheat sheet for Python, covering essential syntax, data structures, and common libraries. Useful for beginners and experienced developers alike.

Basics and Syntax

Variables and Data Types

Variables

Variables are dynamically typed; no explicit declaration needed.

```
x = 5
name = "Alice"
```

Data Types

Common data types: `int`, `float`, `str`, `bool`, `list`, `tuple`, `dict`, `set`

```
age = 30         # int
height = 5.9     # float
message = "Hello" # str
is_active = True  # bool
```

Type Conversion

Convert between data types:

```
str(10)    # Convert int to str
int("20")  # Convert str to int
float(5)   # Convert int to float
```

Operators

Arithmetic

`+` (addition), `-` (subtraction), `*` (multiplication), `/` (division), `//` (floor division), `%` (modulus), `**` (exponentiation)

Comparison

`==` (equal), `!=` (not equal), `>` (greater than), `<` (less than), `>=` (greater than or equal), `<=` (less than or equal)

Logical

`and`, `or`, `not`

Assignment

`=`, `+=`, `-=`, `*=`, `/=`, `%=`, `//=`

Control Flow

If/Elif/Else

```
if condition:
    # Execute if condition is true
elif another_condition:
    # Execute if another_condition is true
else:
    # Execute if no condition is true
```

For Loop

```
for item in iterable:
    # Execute for each item
```

While Loop

```
while condition:
    # Execute while condition is true
```

Break and Continue

```
break    # Exit loop
continue  # Skip current iteration
```

Data Structures

Lists

Creating Lists

```
my_list = [1, 2, 3, 'a', 'b']
```

Accessing Elements

```
my_list[0]  # First element (1)
my_list[-1] # Last element ('b')
my_list[1:3] # Slice [2, 3]
```

List Methods

```
my_list.append(4)   # Add element
my_list.insert(1, 'z') # Insert at index
my_list.remove(1)    # Remove element
my_list.pop(1)       # Remove and return element at index
len(my_list)         # List length
```

Tuples

Creating Tuples

```
my_tuple = (1, 2, 3, 'a', 'b')
```

Tuples are immutable.

Accessing Elements

```
my_tuple[0]  # First element (1)
my_tuple[1:3] # Slice (2, 3)
```

Dictionaries

Creating Dictionaries

```
my_dict = {'name': 'Alice', 'age': 30}
```

Accessing Elements

```
my_dict['name']  # 'Alice'
my_dict.get('age', 0) # 30 (or 0 if not found)
```

Dictionary Methods

```
my_dict['city'] = 'New York'  # Add key-value pair
del my_dict['age']
# Delete key
my_dict.keys()
# Get keys
my_dict.values()
# Get values
my_dict.items()
# Get key-value pairs
```

Sets

Creating Sets	`my_set = {1, 2, 3, 4, 5}` Sets contain unique elements.

Set Methods	`my_set.add(6)  # Add element`
	`my_set.remove(1)  # Remove element`
	`my_set.union({7, 8})  # Union of sets`
	`my_set.intersection({3, 4})  # Intersection of sets`

Functions and Modules

Functions

Defining Functions

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

Calling Functions

```
message = greet("Bob")
print(message) # Output: Hello, Bob!
```

Lambda Functions

```
add = lambda x, y: x + y
result = add(5, 3) # result is 8
```

Default Arguments

```
def power(base, exponent=2):
    return base ** exponent

print(power(3))    # Output: 9
print(power(3, 3)) # Output: 27
```

Modules

Importing Modules

```
import math
print(math.sqrt(25)) # Output: 5.0

from datetime import datetime
print(datetime.now())

import os as operating_system
print(operating_system.getcwd())
```

Common Modules `math`, `datetime`, `os`, `random`, `json`, `re`

File I/O and Error Handling

File I/O

Reading Files

```
with open('my_file.txt', 'r') as f:
    content = f.read()
    print(content)
```

Writing Files

```
with open('my_file.txt', 'w') as f:
    f.write('Hello, file!')
```

Appending to Files

```
with open('my_file.txt', 'a') as f:
    f.write('\nNew line of text.')
```

Error Handling

Try/Except Blocks

```
try:
    result = 10 / 0
except ZeroDivisionError as e:
    print(f"Error: {e}")
finally:
    print("This will always execute.")
```

Raising Exceptions

```
raise ValueError("Invalid input")
```