Python Cheatsheet

1 Variables

Variables store data. Python uses dynamic typing, no need to declare types.

```
name = "Alice"  # String
age = 30  # Integer
pi = 3.14  # Float
```

2 Data Types

Python supports several built-in data types.

```
# Primitives
string = "Hello"  # str
integer = 42  # int
floating = 3.14  # float
boolean = True  # bool
none_value = None  # NoneType

# Collections
list_data = [1, 2, 3]  # list
tuple_data = (1, 2, 3)  # tuple (immutable)
dict_data = {"key": "value"}  # dict
set_data = {1, 2, 3}  # set (unique elements)
```

3 Operators

Operators perform arithmetic, comparison, logical, and other operations.

```
# Arithmetic
sum = 5 + 3  # 8
mod = 10 % 3  # 1

# Comparison
print(5 == 5)  # True
print(5 != 3)  # True

# Logical
and_result = True and False  # False
or_result = True or False  # True
not_result = not True  # False
```

4 Conditionals

Control flow with `if`, `elif`, `else`, or ternary expressions.

```
# If-Elif-Else
age = 18
if age >= 18:
    print("Adult")
elif age >= 13:
```

```
    print("Teen")
else:
    print("Child")

# Ternary
status = "Adult" if age >= 18 else "Minor"
```

5 Loops

Iterate with `for`, `while`, or loop control statements.

```
# For Loop
for i in range(3):
    print(i)  # 0, 1, 2

# While Loop
i = 0
while i < 3:
    print(i)
    i += 1  # 0, 1, 2

# Break and Continue
for i in range(5):
    if i == 3:
        break  # Exit loop
    if i == 1:
        continue  # Skip to next iteration
    print(i)  # 0, 2
```

6 Functions

Functions encapsulate reusable code with optional parameters.

```
# Function Definition
def greet(name="World"):
    return f"Hello, {name}"

print(greet("Alice"))  # Hello, Alice
print(greet())  # Hello, World

# Lambda Function
add = lambda x, y: x + y
print(add(2, 3))  # 5
```

7 List Comprehensions

Concise way to create lists.

```
# Basic List Comprehension
squares = [x**2 for x in range(5)]  # [0, 1, 4, 9, 16]

# With Condition
evens = [x for x in range(10) if x % 2 == 0]  # [0, 2, 4, 6, 8]
```

8 Dictionaries

Key-value pairs for storing data.

```python
person = {"name": "Bob", "age": 30}
print(person["name"])  # Bob

# Add/Update
person["city"] = "New York"

# Dictionary Comprehension
squares_dict = {x: x**2 for x in range(3)}  # {0: 0, 1: 1, 2: 4}
```

9 Sets

Unordered collections of unique elements.

```python
my_set = {1, 2, 3, 3}  # {1, 2, 3}
my_set.add(4)  # {1, 2, 3, 4}
my_set.remove(2)  # {1, 3, 4}

# Set Operations
set_a = {1, 2, 3}
set_b = {2, 3, 4}
print(set_a | set_b)  # Union: {1, 2, 3, 4}
print(set_a & set_b)  # Intersection: {2, 3}
```

10 File Handling

Read from and write to files.

```python
# Writing to a file
with open("example.txt", "w") as file:
    file.write("Hello, Python!")

# Reading from a file
with open("example.txt", "r") as file:
    content = file.read()
    print(content)  # Hello, Python!
```

11 Exception Handling

Handle errors gracefully with `try`, `except`.

```python
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
finally:
    print("This always runs")
```

12 Classes and OOP Basics

Define classes for object-oriented programming.

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        return f"Hi, I'm {self.name}"
```

```
8
9  # Inheritance
10 class Student(Person):
11     def __init__(self, name, age, grade):
12         super().__init__(name, age)
13         self.grade = grade
14
15 bob = Student("Bob", 20, "A")
16 print(bob.greet())  # Hi, I'm Bob
```

13 Standard Libraries

13.1 math

Mathematical functions and constants.

```
1 import math
2 print(math.pi)  # 3.141592653589793
3 print(math.sqrt(16))  # 4.0
4 print(math.factorial(5))  # 120
```

13.2 os

Interact with the operating system.

```
1 import os
2 print(os.getcwd())  # Current working directory
3 os.mkdir("new_folder")  # Create a directory
4 print(os.listdir())  # List directory contents
```

13.3 datetime

Handle dates and times.

```
1 from datetime import datetime
2 now = datetime.now()
3 print(now)  # Current date and time
4 formatted = now.strftime("%Y-%m-%d")  # Format: 2025-05-30
```

14 Notes

- Use `with` for file handling to ensure proper resource management.
- Prefer list comprehensions for concise list creation.

- Use $\texttt{snake}_{c}ase for variable and function names. Leverage standard libraries to avoid reinventing the wheel.$