**Python Beginner Comprehensive Cheat Sheet**

This cheat sheet covers all essential Python concepts for beginners with detailed explanations, properties, and examples.

---

# 1. Basics & Printing

- `print()`: Displays messages or variables.
- Comments: `#` for single-line, `""" ... """` for multi-line.

```
# Print simple text
print("Hello, World!")

# Print multiple items
x = 10
y = 5
print("Sum of", x, "and", y, "is", x+y)
```

---

# 2. Variables & Data Types

- Variables store data of different types.
- Common types:
- int: whole numbers
- float: decimal numbers
- str: text
- bool: True/False

```
x = 10       # integer
pi = 3.14    # float
name = "Alice"  # string
is_student = True  # boolean

# Checking types
print(type(x), type(pi), type(name), type(is_student))
```

**Type Conversion:**

```
int("5")      # converts string to integer
float("3.2")  # converts string to float
str(10)        # converts integer to string
```

## 3. Operators

**Arithmetic Operators:** `+ - * / % // **`

```
a = 10 + 3   # 13
b = 10 ** 2  # 100
c = 10 // 3  # 3 (floor division)
```

**Comparison Operators:** `== != > < >= <=`

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

**Logical Operators:** `and, or, not`

```
x = 5
y = 10
print(x < 10 and y > 5)  # True
print(not x > 10)          # True
```

---

## 4. Input & Output

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))
print(f"Hello {name}, you are {age} years old")
```

---

## 5. Strings

- Strings are sequences of characters.
- Indexing starts at 0.
- Strings are immutable.

```
text = "Python"
print(text[0])   # P
print(text[-1])  # n
print(text[0:4]) # Pyth
```

```
# String methods
print(text.upper())        # PYTHON
print(text.lower())        # python
print(text.replace("P","J"))  # Jython
print(len(text))           # 6
print(text.split("y"))    # ['P', 'thon']
```

---

## 6. Lists

- Ordered, mutable sequences.
- Can contain different types.
- Useful methods: `.append()`, `.insert()`, `.remove()`, `.pop()`, `.sort()`, `.reverse()`

```
nums = [1, 2, 3, 4]
nums.append(5)
nums.insert(2, 10)
nums.remove(3)
last = nums.pop()
print(nums, last)
print(len(nums))
print(nums[1:3])
print(2 in nums)  # True
```

---

## 7. Tuples

- Ordered, immutable sequences.
- Faster and safer than lists for fixed data.
- Can store different types.

```
coords = (10, 20, 30)
print(coords[0])
print(coords[-1])
print(len(coords))
# coords[0] = 5  # ❌ Error
```

---

## 8. Dictionaries

- Key-value pairs.
- Mutable, unordered.
- Access values by key.

```
person = {"name": "Bob", "age": 25}
print(person["name"])
person["city"] = "Mumbai"
print(person.keys())
print(person.values())
print(person.items())
```

---

## 9. Sets

- Unordered collections of unique elements.
- Useful for membership tests and set operations.

```
fruits = {"apple", "banana", "cherry"}
fruits.add("orange")
fruits.remove("banana")

# Set operations
a = {1,2,3}
b = {3,4,5}
print(a | b)  # Union {1,2,3,4,5}
print(a & b)  # Intersection {3}
print(a - b)  # Difference {1,2}
```

---

## 10. If Statements

- Conditional execution.
- `elif` and `else` for multiple cases.

```
x = 10
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")

# One-liner
print("Positive") if x > 0 else print("Non-positive")
```

---

## 11. Loops

- Repeat actions.
- `for` loops: iterate over sequences.
- `while` loops: repeat while condition is True.

```
# For loop
for i in range(5):
    print(i)

# While loop
count = 0
while count < 5:
    print(count)
    count += 1

# Break and Continue
for i in range(5):
    if i == 3: break
    if i == 1: continue
    print(i)
```

---

## 12. Functions

- Reusable blocks of code.
- Can take arguments and return values.
- Lambda functions are anonymous one-line functions.

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

print(greet("Alice"))

# Lambda function
square = lambda x: x**2
print(square(5))  # 25
```

---

## 13. Modules & Libraries

- Modules: pre-built code packages.
- Standard libraries: `math`, `random`, `datetime`, etc.

```python
import math
print(math.sqrt(16))
print(math.pi)

import random
print(random.randint(1,10))

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

---

## 14. File Handling

- Open, read, write files.
- Use `with open()` for safe file handling.

```python
# Write to a file
with open("test.txt", "w") as f:
    f.write("Hello Python")

# Read from a file
with open("test.txt", "r") as f:
    print(f.read())
```

---

## 15. Exception Handling

- Handle errors to prevent program crashes.

```python
try:
    x = int(input("Enter a number: "))
    print(10/x)
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Invalid input")
finally:
    print("Done")
```

---

## 16. List Comprehension

- Create lists concisely.

```
nums = [1,2,3,4,5]
squares = [x**2 for x in nums]
even_nums = [x for x in nums if x%2==0]
print(squares)
print(even_nums)
```

---

## 17. Object-Oriented Programming (OOP)

- Classes are blueprints for objects.
- `__init__` initializes object properties.
- Methods define behavior.

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

    def greet(self):
        print(f"Hello, I am {self.name}")

p1 = Person("Alice", 25)
p1.greet()
print(p1.age)
```

---

## 18. Built-in Functions

- `len()`, `type()`, `sum()`, `min()`, `max()`, `sorted()`, `range()`

```
nums = [1,2,3]
print(len(nums))  # 3
print(sum(nums))   # 6
print(sorted([3,1,2]))  # [1,2,3]
```

---

## 19. Mini Project Ideas

- Number Guessing Game
- Calculator
- To-do list
- Dice rolling simulator
- Text-based adventure game

---

This comprehensive cheat sheet is suitable for detailed revision and hands-on practice.