Python Programming Cheat Sheet (Theory Only)

---

1. Structure of a Python Program

A Python program typically consists of:

- **Shebang Line (Optional)** → #!/usr/bin/env python3 (Used in Unix/Linux scripts).
- **Module Imports** → Importing built-in or custom modules.
- **Function Definitions** → Defined using def.
- **Main Code Execution** → Using if __name__ == "__main__":.

Example Structure:

```
# Import modules
import math

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

# Main Code Execution
if __name__ == "__main__":
    print(greet("Alice"))
```

---

2. Underlying Mechanism of Module Execution

When a Python file is run:

- The **interpreter reads the code** line by line.
- **Modules are imported and executed** before the script runs.
- The special variable __name__ is set to "__main__" when a file is executed directly.

Example: Running a Module vs. Importing

```
# module.py
print("This will always run")

if __name__ == "__main__":
    print("This runs only when executed directly")
```

- Running module.py directly prints both messages.
- Importing module in another file only prints "This will always run".

---

3. Branching and Looping

Branching (Conditional Statements)

if, elif, else → Used for decision-making in Python.

Example:

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

Looping (Iteration)

For Loop → Iterates over sequences (lists, strings, ranges).
While Loop → Runs while a condition is True.
Example:

```
# For Loop
for i in range(3):
    print(f"Iteration {i}")

# While Loop
n = 5
while n > 0:
    print(n)
    n -= 1
```

---

4. Problem Solving Using Branches and Loops

Example: Finding the Largest Number

```
nums = [3, 7, 2, 8, 5]
largest = nums[0]

for num in nums:
    if num > largest:
        largest = num

print("Largest Number:", largest)
```

Example: Checking for Prime Numbers

```
num = 29
is_prime = True

if num > 1:
    for i in range(2, num):
        if num % i == 0:
            is_prime = False
            break

print("Prime" if is_prime else "Not Prime")
```

---

5. Functions in Python

Functions → Reusable blocks of code that can accept input (parameters) and return output.
Example:

```
def add(a, b):
    return a + b

result = add(5, 3)
print("Sum:", result)
```

Default & Keyword Arguments

```
def greet(name="Guest"):
    print(f"Hello, {name}")

greet()  # Uses default value "Guest"
greet("Alice")  # Overrides default
```

Recursive Function Example: Factorial Calculation

```
def factorial(n):
    return 1 if n == 0 else n * factorial(n - 1)
```

```
print(factorial(5))  # Output: 120
```

---

6. Lambda Functions (Anonymous Functions)

Lambda functions are small, anonymous functions defined using lambda.

Syntax: lambda arguments: expression

Example:

```
square = lambda x: x ** 2
print(square(4))  # Output: 16
```

Lambda inside map(), filter(), reduce()

```
nums = [1, 2, 3, 4]
squared = list(map(lambda x: x ** 2, nums))
print(squared)  # Output: [1, 4, 9, 16]
```

---

7. Lists and Mutability

Lists are mutable, meaning elements can be modified.

Example:

```
my_list = [1, 2, 3]
my_list[1] = 10  # Modifies the list
print(my_list)  # Output: [1, 10, 3]
```

List Methods:

- .append(x) → Adds an element.
- .remove(x) → Removes an element.
- .sort() → Sorts the list.
- .reverse() → Reverses the list.

Example: Sorting a List

```
nums = [3, 1, 4, 1, 5]
nums.sort()
print(nums)  # Output: [1, 1, 3, 4, 5]
```

List Comprehensions:

```
squares = [x**2 for x in range(5)]
print(squares)  # Output: [0, 1, 4, 9, 16]
```

---

8. Problem Solving Using Lists & Functions

Example: Finding Even Numbers in a List

```
def find_even(nums):
    return [num for num in nums if num % 2 == 0]

numbers = [1, 2, 3, 4, 5, 6]
print(find_even(numbers))  # Output: [2, 4, 6]
```

Example: Fibonacci Series Using Recursion

```
def fibonacci(n):
    return n if n <= 1 else fibonacci(n-1) + fibonacci(n-2)

print([fibonacci(i) for i in range(6)])  # Output: [0, 1, 1, 2, 3, 5]
```

Key Takeaways

Python Program Structure: Includes imports, functions, and a main section.
Branching & Loops: if-else, for, and while are used for decision-making and iteration.
Functions & Lambda: Functions are reusable, and lambda provides a quick way to define anonymous functions.
Lists & Mutability: Lists allow dynamic modifications and list comprehensions for quick operations.
Problem Solving: Python provides efficient ways to solve problems using lists, loops, and functions.

This **Python Programming Cheat Sheet** covers **program structure, module execution, branching, looping, functions, lambda, lists, and problem-solving techniques**. Let me know if you need further explanations!