Complete Python Programming Handbook

1. Python Basics & Setup

- • Python is high-level, interpreted, dynamically typed language.
- • Install Python and verify using: python --version
- • Run file using: python file.py
- • Interpreter executes code line by line.

2. Variables & Data Types

- • Variables store references to objects.
- • int: Whole numbers
- • float: Decimal numbers
- • str: Text (immutable)
- • bool: True/False
- • type() checks data type.

Example – Variables

```
age = 25
name = 'Nikhil'
price = 99.99
is_active = True
```

3. Type Casting

- • int(), float(), str(), bool()
- • input() always returns string.

Example – Type Casting

```
age = int(input('Enter age: '))
```

4. Operators

- • Arithmetic: + - * / // % **
- • Comparison: == != > < >= <=
- • Logical: and, or, not

5. Conditional Statements

- • if, elif, else
- • Indentation defines block.

Example – Condition

```
marks = 80
if marks >= 75:
    print('Distinction')
else:
    print('Pass')
```

6. Loops

- • for loop (used for sequences)
- • while loop (runs until condition false)
- • break and continue

Example – Loop

```
for i in range(5):
    print(i)
```

7. Data Structures

- • List (mutable)
- • Tuple (immutable)
- • Set (unique values)
- • Dictionary (key-value pairs)

Example – List

```
numbers = [1,2,3]
numbers.append(4)
```

Example – Dictionary

```
person = {'name':'Nikhil', 'age':25}
print(person['name'])
```

8. Functions

- • Defined using def keyword
- • Parameters and return values
- • Default arguments, *args, **kwargs

Example – Function

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

9. OOP (Object Oriented Programming)

- • Class and Object
- • __init__ constructor
- • Inheritance
- • Encapsulation

- • Polymorphism

Example – OOP

```
class Person:
    def __init__(self, name):
        self.name = name
    def greet(self):
        print('Hello', self.name)
```

10. Exception Handling

- • try, except, finally
- • raise custom exceptions

Example – Exception

```
try:
    x = int(input())
except ValueError:
    print('Invalid input')
```

11. File Handling

- • open(), read(), write()
- • Use with statement

Example – File

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

12. Modules & Packages

- • import module
- • Create your own module (.py file)
- • Use if __name__ == '__main__'

Example – Module

```
# mymodule.py
def greet():
    print('Hello')
# main.py
import mymodule
mymodule.greet()
```

13. Virtual Environments

- • python -m venv venv
- • Activate environment
- • Install packages using pip

14. Advanced Topics

- • Decorators
- • Generators (yield)
- • Iterators
- • Lambda functions
- • List comprehensions
- • Multithreading basics
- • Big-O complexity basics

15. Important Interview Questions

- • Difference between list and tuple?
- • Mutable vs immutable?

- • What is GIL?
- • Deep copy vs shallow copy?
- • Explain OOP concepts.
- • How Python manages memory?

16. Best Practices

- • Write clean readable code
- • Use meaningful variable names
- • Follow PEP8 style guide
- • Write modular code
- • Use Git for version control