Introduction to Python 3: A Basic Course Syllabus

Prepared By: Ramesh Pradhan
(pyrameshpradhan@gmail.com)
Backend Engineer
(Chuchuro Firm Pvt. Ltd.)

1. Introduction to Python

- ⇨ About Python - a tool, not a reptile
- ⇨ Features of Python
- ⇨ Why learn Python, and why now?

2. Installation

- ⇨ How to install Python on Windows/ Linux
- ⇨ Python Development Environments (Python interactive shell / Python Editor / Integrated
- ⇨ Development Environment(s) (*IDE*))
- ⇨ Setting Python path / switching between different versions
- ⇨ Introduction to *virtual environments* as a way to manage Python installations

3. The very basics

- ⇨ Python Keywords and Identifiers
- ⇨ Python Statements, Indentation and Comments
- ⇨ Python **I/O** (***input( )*** *and* ***print( )*** *functions*)

4. Python Native Data-Types

- ⇨ Variables
  - i. Numbers (*Integers*, *Floats* and *Complex Numbers*)
  - ii. Strings
- ⇨ Variables assignment/ reassignment
- ⇨ Rules and conventions for variable naming in Python
- ⇨ Python *built-in* methods for numbers and strings
- ⇨ Dynamic types and *type-casting* in Python

5. Basic Operators in Python

- ⇨ Simple mathematical operators:
  - i. *addition, subtraction, multiplication, division, floor division, modulo operator*
- ⇨ Comparison operators *and* Logical / Boolean operators (**and** / **or** / **not**)
- ⇨ Membership operators (**in** / **not in**)
- ⇨ Identity operators (**is**)
- ⇨ Bitwise operators (**>>** / **&** / **cmp** / **^**)
- ⇨ Chaining comparison operators
- ⇨ Evaluation order

6. Python Data-Structures

- ⇨ Python native data-structures
  - i. Lists
  - ii. Tuples
  - iii. Sets and Frozen Sets
  - iv. Dictionary

- ⇨ Manipulating Python data-structures
    i. Indexing and slicing, append/ insert, concatenation, length determination, pop, reverse, duplication, key-value pairs, membership tests etc.
- ⇨ Difference between immutable and mutable data-structures
- ⇨ Using Python operators with Python data-structure
- ⇨ Nested data-structures in Python

7. String Formatting in Python

- ⇨ *C-style* string formatting with placeholders
- ⇨ Escape sequences
- ⇨ ***format( )*** method and formatted string literals (***f-strings***)
- ⇨ alignment, padding, and precision

8. Python Control Statements/ Branching Statements

- ⇨ *Conditionals* and *Boolean* in Python
- ⇨ **if** statement

9. Loops and Iterations

- ⇨ ***for*** loop
- ⇨ ***while*** loop
- ⇨ Looping through *tuple*, *string* and *dictionary*
- ⇨ Python special loops (***for/else***)
- ⇨ Nested loops

10. Break/ Continue and Pass statements

11. **Special (*useful*) operators in Python**

- ⇨ *range*
- ⇨ *enumerate*
- ⇨ *zip*
- ⇨ *reversed*

12. List/ Set/ Dictionary comprehension

13. Functions

- ⇨ The ***def*** statement
- ⇨ doc-strings
- ⇨ Function ***arguments*** and ***return*** values
- ⇨ Assigning default function arguments
- ⇨ Types of function arguments:
    i. default *positional* arguments
    ii. *keyword* arguments (***kwargs***)
    iii. *variable-length* arguments (***args***)
- ⇨ ***Namespace*** and variable ***scope*** (*global*, *local*, *non-local* variables, ***LEGB*** rule)

14. Recursive functions

- ⇨ Python recursive functions
- ⇨ Advantages and disadvantages of recursive functions

15. Single statement blocks and Lambda Functions

- ⇨ Regular functions vs *lambda* functions
- ⇨ Why use *lambda* functions?

16. Python modules and packages

- ⇨ Python package managers (*easy install*, *pip*, *conda*)

- ⇨ Installing python packages using ***pip***
- ⇨ Importing modules
- ⇨ Creating custom modules
- ⇨ Exploring standard library (using ***os*** / ***sys*** module to use underlying system functionality)

17. Python *built-in* functions

- ⇨ Exploring some useful *built-in* functions in Python

18. Python special variables

- ⇨ Magic/ dunder/ special variables and methods
- ⇨ *if __name__ == " __main__"*

19. Python File Objects / File handling

- ⇨ ***Binary*** files *vs* ***text*** files
- ⇨ Opening and closing files in Python
- ⇨ Reading and writing to files (*read* / *write* / *append* modes)
- ⇨ Python file methods
- ⇨ Python directory management (using ***os*** module) / setting file paths
- ⇨ Python ***csv*** module

***20.* Iterators and Generators**

- ⇨ Iterable objects vs *iterators*
- ⇨ Difference between *regular functions* and *generators (yield vs return)*
- ⇨ Advantages of generators over functions

21. Debugging using IDE (*VSCode /PyCharm*)

- ⇨ Configuring the debugger
- ⇨ Python Debug Console
- ⇨ Breakpoints and Logpoints

***22.* Errors and Exception Handling in Python**

- ⇨ Error catching with ***try*** / ***except*** / ***finally***
- ⇨ Raising different kinds of built-in exceptions
- ⇨ User defined exceptions

23. Assertion in Python

- ⇨ Python's ***assert*** syntax
- ⇨ Common pitfalls when using *assert*

24. The *logging* module in Python

- ⇨ Python logging basics
- ⇨ Standard library ***logging*** *module*
- ⇨ Logging levels
- ⇨ Log files

25. Object-Oriented programming in Python

- ⇨ Class and Instances (Objects)
- ⇨ The ***self*** statement
- ⇨ Constructor (***__init__*** method)
- ⇨ Class variables
- ⇨ Class methods / static methods
- ⇨ Sub-classes / Inheritance / Multiple-Inheritance and type hierarchy
- ⇨ Special (magic / dunder) methods
- ⇨ Property decorators (getters / setters / deleters)

- ⇨ Polymorphism / Ducktyping / Operator Overloading / Method Overloading

26. **Python Object serialization (working with parameter files, *JSON*)**
27. **Working with databases in Python**
    - ⇨ Brief introduction to working with ***Structured Databases*** (*SQLite, PostgreSQL*) and ***Unstructured Databases*** (*MongoDB*) using Python