Free

python

Python Fundamentals

Object Oriented Python

We are ready to serve Latest Testing Trends, Are you ready to learn.

New Batches Info

START DATE :

TIMINGS :

DURATION :

TYPE OF BATCH :

FEE :

FACULTY NAME :

What & Why's of Python

1. What is Python?
2. Why do People Use Python?
3. Is Python a "Scripting Language"?
4. Where is Python Used?
5. What Can I do with Python?
6. How is Python Developed and Supported?
7. What are Pythons Technical Strengths?
8. First Steps with Python
9. Some big names and applications using python

Python Fundamentals

Getting Started

1. Getting Started with Python
    a. Python 2 vs Python 3
    b. Python Installation
    c. Python Culture and Zen of Python
    d. Significant Whitespace
    e. Python Standard Library
    f. Python Syntax
2. How Python Runs Programs
    a. Introduction to Python Interpreter
    b. Program Execution
    c. Execution Model Variants
3. How You Run Programs
    a. Interactive Prompt
    b. System Command Lines and Files
    c. Unix Style Executable Scripts
    d. Clicking File Icons
    e. Module Imports and Reloads
    f. Using exec to Run Module Files
    g. The IDLE User Interfaces
    h. Other IDEs
    i. Other Launch Options
    j. Which Option Should I Use?
4. Developer Environment Setup
    a. Visual Studio Code
    b. PyCharm

- Types and **Operators**
  5. Introduction to Python Object Types
     a. Conceptual Hierarchy
     b. Why use Built-in Types?
     c. Python's core Data types
     d. Numbers
     e. Strings
        i. Sequence Operations
        ii. Immutability
        iii. Type-Specific Methods
        iv. Getting Help
        v. Other Ways to Code Strings
        vi. Unicode Strings
        vii. Pattern Matching
     f. Lists
        i. Sequence Operations
        ii. Type Specific Operations
        iii. Bound Checking
        iv. Nesting
        v. Comprehensions
     g. Dictionaries
        i. Mapping Operations
        ii. Nesting
        iii. Missing Keys
        iv. Soring Keys
        v. Iterations & Optimization
     h. Tuples
        i. Why tuples?
     i. Files
     j. Other Core Types
  6. Numeric Types
     a. Basics
     b. Numbers in Action
     c. Other Numeric Types
        i. Decimal
        ii. Fraction
        iii. Sets
        iv. Booleans
     d. Numeric Extensions
  7. Dynamic Typing

a. Missing Declaration Statements
b. Shared References
c. Dynamic Typing Is Everywhere

8. String Fundamentals
    a. String Basics
    b. String Literals
        i. Single, Double, Triple Quotes
        ii. Escape Sequences
    c. Strings in Action
        i. Basic Operations
        ii. Indexing and Slicing
        iii. String Conversion Tools
    d. String Methods
    e. String Formatting Expressions and Methods
9. Lists and Dictionaries
    a. Lists
    b. Lists in Action
    c. Dictionaries
    d. Dictionaries in Action
10. Tuples, Files
    a. Tuples
        i. Tuples in Action
    b. Files
        i. Opening Files
        ii. Using Files
        iii. Files in Action
        iv. Text and Binary Files
        v. Storing Python Objects in Files: Conversions
        vi. Storing Native Python Objects: pickle
        vii. Storing Python Objects in JSON Format
        viii. File Context Mangers
        ix. Other File Tools
    c. Core Types Review & Summary
        i. Object Flexibility
        ii. References vs Copies
        iii. Comparisons, Equality and Truth
        iv. Meaning of True and False in Python
        v. Python Type Hierarchies
        vi. Type Objects
        vii. Other Types in Python

d. Built in Type Gotchas

11. Effective Python (Python Best Practices)
    a. Pythonic Thinking
    b. Lists and Dictionaries

Statements & Syntax

12. Introducing Python Statements
    a. Pythons Conceptual Hierarchy
    b. Python Statements
    c. Why Indentation Syntax?
    d. Special Cases
    e. Interactive Loops
13. Assignments, Expressions, and Prints
    a. Assignments Statements
    b. Expression Statements
    c. Print Operations
14. If Tests and Syntax Rules
    a. If Statements
    b. Python Syntax
    c. Truth Values and Boolean Tests
    d. The if/else Ternary Expression
15. While and for loops
    a. While loops
    b. Break, continue, pass and the Loop else
    c. For loops
    d. Loop Coding Techniques
16. Iterations and Comprehensions
    a. Iterations
    b. List Comprehensions
    c. Other Iteration Contexts
    d. Iterables
        i. Range
        ii. Map, zip and filter Iterables
        iii. Multiple vs Single Pass Iterators
        iv. Dictionary View Iterables
    e. Other Iteration Topics
17. Debugging Python
    a. pdb
    b. IDE Debugging
18. Documentation
    a. Python Documentation Sources

i. # Comments
ii. The dir Function
iii. DocStrings: __doc__
iv. PyDoc: The help Function
v. PyDoc: HTML Reports
vi. Beyond docstrings: Sphinx

- **Functions and Generators**

19. Function Basics
    a. Why Use Functions?
    b. Coding Functions
    c. Definitions & Calls
    d. Intersecting Sequences
20. Scopes
    a. Python Scope Basics
        i. Scope Details
        ii. Name Resolution: The LEGB Rule
        iii. Built-in Scope
    b. The global Statement
    c. Scopes and Nested Functions
    d. The nonlocal Statement
    e. Why nonlocal?
21. Arguments
    a. Argument Passing Basics
    b. Special Argument-Matching Modes
    c. Generalized Set Functions
22. Advanced Function Topics
    a. Function Design Concepts
    b. Recursive Functions
    c. Function Objects: Attributes and Annotations
    d. Anonymous Functions: lambda
    e. Functional Programming tools (map, filter and reduce)
23. Comprehensions & Generations
    a. List Comprehensions and Functional Tools
    b. Generator Functions and Expressions
    c. Comprehension Syntax Summary
24. Benchmarking
    a. Timing Iteration Alternatives
    b. Timing Iterations and Pythons with timeit
    c. Pystones
    d. Function Gotchas

25. Effective Python (Python Best Practices)
    a. Functions
    b. Comprehensions and Generators

Modules and Packages

26. Modules: The Big Picture
    a. Why use modules?
    b. Python Program Architecture
    c. How Imports Work
    d. Byte Code Files: __pycache__
    e. Module Search Path
27. Module Coding Basics
    a. Module Creation
    b. Module Usage
    c. Module Namespaces
    d. Reloading Modules
28. Module Packages
    a. Package Import Basics
    b. Why Use Package Imports?
    c. Python Relative Imports
    d. Namespace Imports
29. Advanced Module Topics
    a. Module Design Concepts
    b. Data Hiding in Modules
    c. Enable Future Language Features: __future__
    d. Mixed Usage Modes: __name__ and __main__
    e. Changing Module Search Path
    f. The as Extension for import and from
    g. Importing Modules by Name String

Object Oriented Python

Classes and Object-Oriented Programming

1. Object Oriented Programming: Big Picture
    a. Why Use Classes
    b. OOP high level overview
        i. Classes and Instances
        ii. Method Calls
        iii. Coding Class Trees
        iv. Operator Overloading
2. Class Coding Basics
    a. Generating Multiple Instance Objects

   b. Inheritance
   c. Operator Overloading
   d. Realistic Example
      i. Making Instances
      ii. Adding Behavior Methods
      iii. Operator Overloading
      iv. Customizing Behavior by Sub-classing
      v. Customizing Constructors
      vi. Using Introspection tools
      vii. Storing Objects in Database (Pickles and Shelves)
3. Class Coding Details
   a. The class Statement
   b. Methods
   c. Inheritance
   d. Namespaces
   e. Documentation Strings Revisited
   f. Classes vs Modules
4. Designing with Classes
   a. Python and OOP
   b. OOP and Inheritance
   c. OOP and Composition
   d. OOP and Delegation
   e. Multiple Inheritance
5. Advanced Class Topics
   a. Extending Built-in Types
   b. Static and Class Methods
   c. Decorators and Metaclasses
   d. The super Built-in Function
6. Exception Basics
   a. Why Use Exceptions
   b. Exceptions:
      i. Default Exception Handler
      ii. Catching Exceptions
      iii. Raising Exceptions
      iv. User-Defined Exceptions
      v. Termination Actions
7. Exception Coding Details
   a. The try/except/else Statement
   b. The try/finally Statement
   c. The raise Statement

   d. The assert Statement
   e. With/as Context Managers
8. Exception Objects
   a. Exception Hierarchies
   b. Built-In Exception Classes
   c. Nesting Exception Handlers
   d. Exception Idioms
   e. Exception Design Tips
9. Managed Attributes
   a. Why Manage Attributes
   b. Properties
   c. Descriptors
   d. __getattr__ and __getattribute__
10. Decorators
   a. What's the Decorator?
   b. Basics
      i. Function Decorators
      ii. Class Decorators
      iii. Decorator Nesting
      iv. Decorator Arguments
      v. Decorators Manage Functions and Classes, Too
   c. Coding Function Decorators
   d. Coding Class Decorators
   e. Managing Functions and Classes Directly
11. Metaclasses
   a. Metaclass Model
   b. Declaring Metaclasses
   c. Coding Metaclasses
   d. Metaclass vs Superclass
   e. Metaclass Methods
12. Logging and Tracing
   a. Logger Objects
   b. Logger Levels
   c. Formatter
   d. Filter Objects
   e. Log Record Objects and Attributes
   f. Logger Adapter Objects
13. Concurrency
   a. Starting and Stopping Threads
   b. Thread Communications

c. Locking Critical Sections
d. Locking with Deadlock Avoidance
e. Storing Thread Specific State
f. Creating a Thread Pool
g. Performing a d Simple Parallel Programming
h. Dealing with GIL
i. Defining an Actor Task
j. Implementing Publish/Subscribe Messaging
k. Polling Multiple Thread Queues

14. Effective Python (Best Practices)
    a. Classes and Interfaces
    b. Metaclasses and Attributes
    c. Concurrency and Parallelism
    d. Robustness and Performance
    e. Testing and Debugging
    f. Collaboration