**About Python**

*Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.*

***Why PYTHON***

1. [*Python’s popularity & high salary*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#popularity&HighSalary)
2. [*Python is used in Data Science*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#DataScience)
3. [*Python’s scripting & automation*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#scripting)
4. [*Python used with Big Data*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#BigData)
5. [*Python supports Testing*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#testing)
6. [*Computer Graphics in Python*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#ComputerGraphics)
7. [*Python used in Artificial Intelligence*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#ArtificialIntelligence)
8. [*Python in Web Development*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#WebDevelopment)
9. [*Python is portable & extensible*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#Portable)
10. [*Python is simple & easy to learn*](https://www.edureka.co/blog/10-reasons-why-you-should-learn-python#Simple&Easy)

**Who can do PYTHON**

* *Absolute Beginners.*
* *Previous System C-C++ and Java knowledge would be added advantage*

**Prerequisites**

***No Such prerequisites to learn python but if you have basic knowledge of any programming then you can grasp python even faster.***

* ***Knowledge of Programming***

**Program Objectives**

* *Practical based learning.*
* *Individual should be able to grab the relevant opportunity.*
* *Individual will be able to work on the project immediately.*

**Program Contents PYTHON**

*Basic Python :*

*1. Using the Python Interpreter*

*2.1. Invoking the Interpreter*

*2.1.1. Argument Passing*

*2.1.2. Interactive Mode*

*2. The Interpreter and Its Environment*

*2.2.1. Source Code Encoding*

*3. An Informal Introduction to Python*

*3.1. Using Python as a Calculator*

*3.1.1. Numbers*

*3.1.2. Strings*

*3.1.3. Unicode Strings*

*3.1.4. Lists*

*3.2. First Steps Towards Programming*

*4. More Control Flow Tools*

*4.1. if Statements*

*4.2. for Statements*

*4.3. The range() Function*

*4.4. break and continue Statements, and else Clauses on Loops*

*4.5. pass Statements*

*4.6. Defining Functions*

*4.7. More on Defining Functions*

*4.7.1. Default Argument Values*

*4.7.2. Keyword Arguments*

*4.7.3. Arbitrary Argument Lists*

*4.7.4. Unpacking Argument Lists*

*4.7.5. Lambda Expressions*

*4.7.6. Documentation Strings*

*5. Data Structures*

*5.1. More on Lists*

*5.1.1. Using Lists as Stacks*

*5.1.2. Using Lists as Queues*

*5.1.3. Functional Programming Tools*

*5.1.4. List Comprehensions*

*5.1.4.1. Nested List Comprehensions*

*5.2. The del statement*

*5.3. Tuples and Sequences*

*5.4. Sets*

*5.5. Dictionaries*

*5.6. Looping Techniques*

*5.7. More on Conditions*

*5.8. Comparing Sequences and Other Types*

*6. Modules*

*6.1. More on Modules*

*6.1.1. Executing modules as scripts*

*6.1.2. The Module Search Path*

*6.1.3. “Compiled” Python files*

*6.2. Standard Modules*

*6.3. The dir() Function*

*Advance Python :*

*6.4. Packages*

*6.4.1. Importing \* From a Package*

*6.4.2. Intra-package References*

*6.4.3. Packages in Multiple Directories*

*7. Input and Output*

*7.1. Fancier Output Formatting*

*7.1.1. Old string formatting*

*7.2. Reading and Writing Files*

*7.2.1. Methods of File Objects*

*7.2.2. Saving structured data with json*

*8. Errors and Exceptions*

*8.1. Syntax Errors*

*8.2. Exceptions*

*8.3. Handling Exceptions*

*8.4. Raising Exceptions*

*8.5. User-defined Exceptions*

*8.6. Defining Clean-up Actions*

*8.7. Predefined Clean-up Actions*

*9. Classes*

*9.1. A Word About Names and Objects*

*9.2. Python Scopes and Namespaces*

*9.3. A First Look at Classes*

*9.3.1. Class Definition Syntax*

*9.3.2. Class Objects*

*9.3.3. Instance Objects*

*9.3.4. Method Objects*

*9.3.5. Class and Instance Variables*

*9.4. Random Remarks*

*9.5. Inheritance*

*9.5.1. Multiple Inheritance*

*9.6. Private Variables and Class-local References*

*9.7. Odds and Ends*

*9.8. Exceptions Are Classes Too*

*9.9. Iterators*

*9.10. Generators*

*9.11. Generator Expressions*

*10. Brief Tour of the Standard Library*

*10.1. Operating System Interface*

*10.2. File Wildcards*

*10.3. Command Line Arguments*

*10.4. Error Output Redirection and Program Termination*

*10.5. String Pattern Matching*

*10.6. Mathematics*

*10.7. Internet Access*

*10.8. Dates and Times*

*10.9. Data Compression*

*10.10. Performance Measurement*

*10.11. Quality Control*

*10.12. Batteries Included*

*11. Brief Tour of the Standard Library – Part II*

*11.1. Output Formatting*

*11.2. Templating*

*11.3. Working with Binary Data Record Layouts*

*11.4. Multi-threading*

*11.5. Logging*

*11.6. Weak References*

*11.7. Tools for Working with Lists*

*11.8. Decimal Floating Point Arithmetic*

*12. What Now?*

*13. Interactive Input Editing and History Substitution*

*13.1. Line Editing*

*13.2. History Substitution*

*13.3. Key Bindings*

*13.4. Alternatives to the Interactive Interpreter*

*14. Floating Point Arithmetic: Issues and Limitations*

*14.1. Representation Error*