

Python

BccFalna.com

097994-55505

Kuldeep Chand

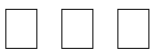
With this eBook you can Learn Programming Fundamentals with Deep Details in easy to understand Hindi Language.

I have Included so many Example Programs and Code Fragments in this ebook to easily understand various kinds of Programming Concept with Detailed Program Flow Discussion to understand the working of the Program Step by Step.

Without learning "Python" Language, you can't learn Machine Learning (ML), Artificial Intelligence (AI), Big Data, Data Science, Robotics, etc..., properly because all these could be learnt very easily if you know Python.

Python

in Hindi



Kuldeep Chand

Python in HINDI

Copyright © on 2020 by Kuldeep Chand

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editors: **Kuldeep Chand**

Distributed to the book trade worldwide by BccFalna.com, Subhash Road, Near Vidhya Jyoti School, Falna Station Dist. Pali (Raj.) Pin 306116

e-mail bccfalna@gmail.com,

or

visit <https://www.bccfalna.com>

For information on translations, please contact BccFalna.com, Subhash Road, Near Vidhya Jyoti School, Falna Station Dist. Pali (Raj.) Pin 306116

Mob.: +91-9799455505

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, the author shall not have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this book.

This book is dedicated to those

who really wants to be

a

PROFESSIONAL DEVELOPER

INDEX
OF
CONTENTS

Index of Contents

Python – The Introduction	14
Advantages of Python	14
<i>Python is an Scripting Language</i>	14
<i>Python is High Performing Scripting Language.....</i>	15
<i>Python is Multi-Purpose Scripting Language</i>	16
<i>Python is Developer Friendly Scripting Language</i>	17
<i>Python is Portable Scripting Language</i>	18
<i>Python have Large Collection of Supports</i>	18
<i>Python Supports Multiple Types of Component Integration</i>	19
Disadvantages of Python	19
Python Popularity	21
Where is Used Python?	22
<i>System Programming.....</i>	23
<i>Graphical User Interfaces – GUIs.....</i>	24
<i>Internet and Network Programming.....</i>	25
<i>Component Integration.....</i>	27
<i>Database Programming</i>	27
<i>Rapid Prototyping.....</i>	29
<i>Scientific and Numeric Programming.....</i>	29
<i>Designing, Image Processing and Game Programming.....</i>	30
<i>Hardware, System and Embedded Programming.....</i>	30
<i>Document and Data Processing Programming</i>	31
Python Features for Developer?	31
<i>Python is Object Oriented, Procedural and Functional.....</i>	32
<i>Free and Open Source</i>	32
<i>Python is Portable.....</i>	33
<i>Python is Powerful.....</i>	34
Dynamically Typed.....	34
Automatic Dynamic Memory Management	34
Built-In Support for Large-Scale Application Development.....	35
Built-In Data-Structure Object Types	36
Data-Structure Object Types Built-In Access and Manipulation Support.....	36
Standard Core-Libraries Support.....	37
Third-Party External Libraries Support.....	37
<i>Python is Multi-Language Supported.....</i>	37

<i>Python is Easy to Learn and Use</i>	38
Python v/s Others	38
<i>Python – Interpreter Internal Working</i>	41
Python Interpreter	41
<i> IDLE = Integrated Development and Learning Environment</i>	42
<i> Python Shell in Command Line</i>	44
<i> Text Editors</i>	45
<i> Interactive Prompt and Python Shell</i>	46
Only Python Commands in Interactive Prompt.....	48
No print() Statement in Interactive Prompt.....	48
No Indentation in Interactive Prompt or Python Script File.....	49
Only One Statement Runs at a Time in Interactive Prompt	49
Multiline Compound Statements in Interactive Prompt.....	49
<i> Linux/Unix Executable Scripts</i>	51
<i> Modules – The Python Script Files</i>	54
Importing and Reloading Module.....	54
Module Attributes.....	59
Modules and Namespaces	61
Internal Working of Python	63
<i> Byte Code Compilation</i>	63
<i> Python Virtual Machine (PVM)</i>	66
Python Implementaion Alternatives	69
<i> CPython – The Standard</i>	70
<i> Jython – Python for Java</i>	70
<i> IronPython – Python for .NET</i>	71
<i> Stackless – Python for Concurrency</i>	72
<i> PyPy – Python for Speed</i>	73
Frozen Binaries – The Python Executable Package	74
<i>Python – Object Types</i>	77
Python Program Hierarchy	77
Benefits of Built-In Types	78
Core Data Types	79
Identifiers	82

Numbers	84
<i>Python Comment.....</i>	<i>86</i>
<i>Python print() Function.....</i>	<i>87</i>
<i>Type Conversion or Type Casting.....</i>	<i>88</i>
<i>Complex Mathematical Operations.....</i>	<i>92</i>
String	92
<i>Identifier – Variable.....</i>	<i>94</i>
<i>Negative Indexing.....</i>	<i>94</i>
<i>Slicing.....</i>	<i>95</i>
<i>Immutability - Unchangeability.....</i>	<i>98</i>
join() Method.....	100
bytearray() Function	102
<i>Type-Specific Methods.....</i>	<i>105</i>
Method Chaining in Python.....	108
String Formatting.....	109
dir() Function	111
help() Function.....	113
<i>String Coding Other Ways.....</i>	<i>114</i>
<i>Unicode Supported Strings.....</i>	<i>117</i>
Lists	119
<i>Sequence Operations.....</i>	<i>119</i>
<i>Type-Specific Operations.....</i>	<i>121</i>
<i>Bound Checking.....</i>	<i>123</i>
<i>Nesting.....</i>	<i>125</i>
<i>Comprehensions.....</i>	<i>126</i>
Dictionaries	128
<i>Mapping Operations.....</i>	<i>129</i>
<i>Nesting.....</i>	<i>133</i>
<i>Workin with Non-Existing Keys.....</i>	<i>134</i>
<i>Sorting the Dictionary.....</i>	<i>138</i>
<i>Iteration and Optimization.....</i>	<i>143</i>
Tuples.....	146
Files	147
<i>Binary Bytes Files.....</i>	<i>153</i>
<i>Unicode Text Files.....</i>	<i>154</i>
Other Core Types.....	154
<i>Boolean Core Types.....</i>	<i>157</i>

<i>Type Testing</i>	159
<i>User-Defined Data Types or User Defined Class</i>	163
<i>Everthing Else is Related to Program Execution</i>	165
Python – Numeric Types	167
Numeric Type Fundaments	167
<i>Numerical Literals</i>	168
Built-In Numerical Tools	172
<i>Python Expression Operators</i>	173
Operator Precedence.....	175
Automatic Type Conversion – Type Casting.....	177
Manual or Explicit Type Conversion – Type Casting	178
Operator Overloading and Polymorphism.....	179
<i>Working with Numbers</i>	180
Python Variables.....	180
Normal Comparision and Chained Comparision.....	183
Classic Division and Floor Division	186
Complex Numbers.....	187
Bitwise Operators.....	188
<i>Built-In Methemathical Functions</i>	191
<i>Other Numeric Types</i>	194
Decimal Type	194
Fraction Type	196
Sets	197
Booleans	204
Python – Dynamic Typing	207
Variables, Objects and References	207
<i>Variable Creation</i>	207
<i>Variable Types</i>	207
<i>Use of Variable</i>	208
Header Fields - Designator and Reference Counter	210
Shared References	213
<i>Shared Reference and In-Place Change</i>	216
<i>Shared Reference and Equality Check</i>	220

Python – String Fundamentals.....	224
Unicode.....	224
String Basics.....	225
String Literals.....	226
<i>Single Quotes and Double Quotes.....</i>	<i>226</i>
<i>Escape Sequence Character Constants.....</i>	<i>228</i>
<i>Treat Escape Sequence Character as Raw String.....</i>	<i>229</i>
<i>Triple Quotes Code Multiline Block Strings.....</i>	<i>231</i>
String Actions.....	232
<i>Indexing and Slicing.....</i>	<i>234</i>
<i>String Conversion.....</i>	<i>236</i>
String Methods.....	239
<i>Find and Replace.....</i>	<i>240</i>
<i>split() Method.....</i>	<i>244</i>
String Formatting.....	245
<i>String Formatting Expression.....</i>	<i>246</i>
<i>String Formatting Method Call.....</i>	<i>252</i>
Python – List and Dictionary.....	256
Lists.....	256
<i>Basic Operations.....</i>	<i>257</i>
<i>Common Methods.....</i>	<i>263</i>
Dictionaries.....	269
<i>Dictionary Use Cases.....</i>	<i>277</i>
Python – Tuples and Files.....	285
Tuples.....	285
Files.....	292
<i>Text and Binary Files.....</i>	<i>302</i>
<i>Writing and Reading Python Objects.....</i>	<i>305</i>
<i>Writing and Reading Native Python Objects Using Pickle.....</i>	<i>308</i>
<i>Writing and Reading Python Objects in JSON Format.....</i>	<i>310</i>
Python – Assignment Statements.....	317

Basic Rules of Creating Python Program	318
Assignment Statements.....	321
Shorthand Assignment Statements	327
Expression Assignment Statements	330
Print Statement	331
<i>Python – Conditional Statements</i>	<i>338</i>
if, if ... else and if ... elif ... else Statements.....	338
Block Statements Delimiters.....	346
Truth Values and Boolean Tests	348
Turnery Expression – The Shorthand of if ... else.....	352
<i>Python – Looping and Iteration Protocol</i>	<i>354</i>
while Loop.....	354
break, continue, pass and Loop else	360
for Loop	365
Nested Loop.....	370
Loops and File Handling.....	373
Counter Loop	376
Iteration Protocol.....	378
Manual Iterations.....	379
List Comprehension	386
<i>Python – Function Basics</i>	<i>395</i>
def Statement for Defining Functions.....	399
Python Polymorphism	403
Local Variables and Statement Block.....	406
Python Scope Basics.....	406
Name Resolution Scheme – The LEGB Rule.....	413
Built – In Scope.....	418
Global Scope.....	421
Minimize Global Variable Names.....	424
Minimize Cross-Module Variable Value Modification.....	425

Enclosing Scope – Nested Functions.....	427
Closures – The Factory Function.....	429
Scopes with Lambda Expression.....	433
nonlocal Statement.....	439
<i>Python – Function Advanced</i>	447
Shared References, Arguments and Return Values	448
Arguments Matching and Default Arguments	452
Keyword-Only Arguments.....	465
Recursive Function	470
Function Object Attributes.....	474
Function Object Annotations.....	480
lambda Expression – The Anonymous Functions	482
Functional Programming Tools	489
<i>Python – Comprehensions and Generations.....</i>	499
List Comprehension and Function Tools	499
<i>List Comprehension and Map.....</i>	<i>500</i>
<i>List Comprehension - Testing and Nesting.....</i>	<i>503</i>
<i>List Comprehension Syntax.....</i>	<i>505</i>
<i>List Comprehensions and Matrixes.....</i>	<i>507</i>
Generator Functions and Expressions.....	511
<i>Generator Functions.....</i>	<i>511</i>
yield vs return Statement.....	511
<i>Generator Expressions</i>	<i>516</i>
Generator Expressions - Iterable and Comprehension	518
Set and Dictionary Comprehension	523
<i>Scopes and Comprehension Variables.....</i>	<i>524</i>
<i>Comprehending Set and Dictionary Comprehensions.....</i>	<i>526</i>
<i>Python – Modules Basics</i>	529
Why Use Modules?	530
Code Reuse	530
System Namespace Partitioning.....	530

<i>Implementing Shared Services or Shared Data.....</i>	<i>531</i>
Simple Architecture of Python Program	531
<i>Imports and Attributes.....</i>	<i>532</i>
<i>Standard Library Modules.....</i>	<i>535</i>
<i>Working of import Statement.....</i>	<i>536</i>
Fine the Module to be Loaded	536
Compile the Module if Not Already Compiled.....	537
Run the Loaded Module.....	538
Module Search Path of Python.....	539
<i>sys.path List.....</i>	<i>543</i>
<i>Module File Selection.....</i>	<i>545</i>
Import and from Statements	547
Module Namespaces.....	561
Modules Reloading	567
<i>Python – Module Packages</i>	<i>572</i>
Package Import	572
<i>Package Initialization.....</i>	<i>575</i>
<i>Module Usability Declaration.....</i>	<i>575</i>
<i>Module Namespace Initialization.....</i>	<i>576</i>
<i>from ... * Statement Behavior.....</i>	<i>576</i>
Why Package Import than Module Import.....	580
Package Relative Imports.....	585
Python 3.3+ Namespace Packages	588
Module Design Concepts.....	594
Data Hiding in Modules.....	595
<code>__future__</code> – Enabling Future Language Features.....	598
<code>__name__</code> and <code>__main__</code> Usage Modes	599
<i>Last but not Least. There is more...</i>	<i>601</i>

PYTHON

Introduction

Python – The Introduction

अगर आप पहली बार Programming सीख रहे हैं और Python आपकी First Programming Language है, तो आपने Python से शुरूआत करके कोई गलती नहीं कर रहे हैं। और अगर Python आपकी First Programming Language नहीं है, तो निश्चित रूप से आप जानते हैं कि आप क्या और क्यों सीखने जा रहे हैं और Python आपके लिए सीखना क्यों जरूरी है।

अगर केवल इतना कहा जाए कि यदि आप सिर्फ Python को ठीक से सीख लें, तो आपको किसी भी तरह की Programming से सम्बंधित जरूरत को पूरा करने के लिए किसी भी अन्य Programming Language को सीखने की जरूरत नहीं रहेगी, तो ये बात बिल्कुल भी गलत नहीं होगी।

क्योंकि Python अपने आप में इतना विस्तृत और बहुआयामी विषय है, जिसे पूरी तरह से सीखने में हफ्ते, महीने नहीं, बल्कि सालों का समय भी कम पड़ सकता है और इसी तथ्य को हम इस Chapter में थोड़ा विस्तार से समझने की कोशिश करेंगे कि क्यों सीखना चाहिए Python को और किन-किन तरह की जरूरतों को आसानी से पूरा किया सकता है Python द्वारा।

Advantages of Python

हालांकि वर्तमान में ढेर सारी अन्य आधुनिक Programming Languages हैं, जो कि लम्बे समय से विभिन्न प्रकार की जरूरतों को पूरा करने के लिए Use हो रही हैं। इसलिए किसी के भी मन में ये सवाल पैदा हो सकता है कि आखिर Python को क्यों सीखा जाए जबकि ढेर सारी अन्य Modern Programming Languages पहले से ही सफलतापूर्वक उपयोग में ली जा रही हैं।

इस सवाल का जवाब ठीक से समझने के लिए हमें Python के Advantages को समझना होगा। इसलिए, सबसे पहले हम Python की उन विशेषताओं को ही देख लेते हैं।

Python is an Scripting Language

Python एक General Purpose Programming Language है जो Procedural, Functional व Object Oriented Paradigm तीनों को समान रूप से और पूरी Capabilities के साथ Support करता है।

सामान्यतः लोग Scripting Language व Programming Language को अलग-अलग समझते हैं जो कि आंशिक रूप से सही भी है लेकिन अक्सर लोग किसी Scripting Language को Programming Language की तुलना में कमजोर व कम उपयोग समझ लेते हैं, जो कि पूरी तरह से गलत है।

वास्तव में एक Programming Language व Scripting Language में केवल मात्र इतना ही अन्तर होता है कि Programming Language के Codes, एक ही बार में Compile हो कर Machine Language Codes में Convert हो जाते हैं, जिसकी Current Operating System पर आधारित एक अलग Executable File बन जाती है परिणामस्वरूप हम जब भी कभी इन Compiled Code वाले Programs को Run करते हैं, हमें इनकी Source Code File की जरूरत नहीं रह जाती।

लेकिन जिन Programs को हम किसी Scripting Language का प्रयोग करते हुए Develop करते हैं, वे हमेशा Source Codes के रूप में ही रहते हैं और इन Programs को हमें जब भी कभी Run करना होता है, हमें उस Language के Interpreter की जरूरत पड़ती है, जिसका प्रयोग करते हुए हमने उस Scripting Language के Program को Develop किया है क्योंकि ये Interpreter ही उस Program के Source Codes को Machine Language Code में Convert करता है जो कि पूरी तरह से RAM में रहता है और जैसे ही हम उस Scripting Language के Program को Terminate कर देते हैं, हमारा Machine Code भी पूरी हमारे Computer की Memory से पूरी तरह से Destroy हो जाता है।

इस प्रकार से Programming Language व Scripting Language में केवल इतना ही अन्तर है कि Programming Language के Compiled Codes की एक Separate Executable File बन जाती है जिसके लिए फिर से Source Code File की जरूरत नहीं रहती। जबकि Scripting Language के Source Codes को एक Interpreter द्वारा हर बार Machine Codes में Translate करके Execute किया जाता है।

अतः यदि सरलतम शब्दों में कहें, तो प्रत्येक Scripting Language अपने आप में एक Complete Programming Language ही होती है और एक Scripting Language द्वारा हर उस जरूरत को पूरा किया जा सकता है, जिसे पूरा करने के लिए हम किसी Compiled Language को Use कर सकते हैं।

Python, PHP, Perl, JavaScript, TypeScript, ActionScript, CoffeeScript, Ruby, ASP, JSP, VBScript, Tcl, Lua, Unix Shell Scripts (*ksh, csh, bash, sh, etc...*), Windows Batch Script, PowerShell आदि सभी Scripting Languages भले ही कहलाती हों, लेकिन ये सभी अपने आप में पूर्ण Programming Languages हैं और इन्हें किसी भी Compiled Language से कम समझने की गलती न करें।

Python is High Performing Scripting Language

Python एक ऐसी Language है, जो C/C++ जैसी क्षमताओं से युक्त है और लगभग C/C++ जितनी ही Powerful है।

इसलिए न केवल High Level Application Software से सम्बंधित जरूरतों को पूरा करने के लिए बल्कि Machine Dependent Low Level Application Software से सम्बंधित जरूरतों को भी आसानी से पूरा करने के लिए Python को बहुत ही प्रभावशाली तरीके से Use किया जा सकता है और इसीलिए Python, वर्तमान समय की सबसे ज्यादा तेज गति से Use होने वाली Programming Language बन चुकी है।

C/C++ के बाद अगर System Level Programming के लिए किसी एक Language पर भरोसा किया जा सकता है, तो वो Python ही है क्योंकि Python, C/C++ Code Libraries को बड़ी ही आसानी से Reuse कर सकता है और High Performance Result दे सकता है और ऐसा इसीलिए है क्योंकि Python, C/C++ के बाद अस्तित्व में आया, इसलिए इसे शुरू से ही C/C++ के Compatible रखा जाना जरूरी था जो कि अपने समय की ही नहीं, बल्कि वर्तमान की भी सबसे ज्यादा Powerful Compiled Programming Languages हैं और आज भी जहां High Performance की जरूरत होती है, C/C++ का कोई भी अन्य Alternative नहीं है।

लेकिन अक्सर जब Performance के साथ Fast Development Speed की भी जरूरत होती है, तब एकमात्र Python ही C/C++ के Best Possible Alternative का Role Play करने में पूरी तरह से सक्षम हो पाता है।

Python is Multi-Purpose Scripting Language

Python की सबसे बड़ी विशेषता ये है कि Python Language के Programming Codes की Readability काफी अच्छी होती है क्योंकि Python को बनाया ही इसी तरह से गया है ताकि इसके Program Codes आसानी से Readable रहें। साथ ही किसी भी अन्य Scripting Language की तुलना में कहीं ज्यादा आसानी से Reusable व Maintainable भी रहें।

Python Scripting के Syntaxes काफी आसानी से याद हो जाते हैं। क्योंकि विभिन्न प्रकार के Data Structures को लगभग समान प्रकार के Python Codes द्वारा Specify किया जाता है। यानी Dictionary Create करने के लिए लिखे गए Code और Tuple Create करने के लिए लिखे गए Code में केवल Curly Braces व Bracket का ही अन्तर होता है।

इसीलिए यदि आप Dictionary Create व Manipulate करने से सम्बंधित Codes को समझ लें, तो आपने काफी हद तक Tuple को Create व Manipulate करने का Code भी समझ ही लिया होता है क्योंकि दोनों ही Data Structures को Access व Manipulate करने से सम्बंधित सभी Functionalities लगभग एक समान ही हैं।

इसी वजह से Python को सीखना, समझना और याद रखना किसी भी अन्य Programming Language को सीखने की तुलना में कहीं ज्यादा आसान व Fast होता है और एक बार Python को सीख लेने के बाद तरह-तरह की जरूरतों को पूरा करने के लिए बार-बार Python के Manual को नहीं देखना पड़ता।

Python को Develop करते समय इसके Creators के दिमाग में यही बात थी कि एक ही काम को करने के लिए चाहे जितने भी तरीके हों, लेकिन सभी तरीकों में लिखे जाने वाले Codes में एकरूपता रहनी चाहिए। ताकि किसी Task को Complete करने के लिए चाहे जो भी तरीका अपनाया गया हो, लेकिन सभी तरीके एक दूसरे के समरूप हों ताकि कोई भी Programmer किसी दूसरे Programmer द्वारा लिखे गए Codes को आसानी से पढ़ व समझ सके।

अन्य शब्दों में कहें तो Python में किसी Task को Complete करने का एक निश्चित तरीका होता है जो कि Most Obvious तरीका होता है। फिर उसी Task को Complete करने के कुछ अन्य Alternative तरीके होते हैं जो छोटे-मोटे Changes के साथ उस पहले तरीके को ही Follow कर रहे होते हैं।

एक ध्यान रखने वाली बात ये है कि Python हमारे लिए किसी भी तरह के Automated Decisions नहीं लेता जैसाकि अन्य Modern Programming Languages में होता है। उदाहरण के लिए कई Programming Languages में Double Quotes के बीच कुछ भी न लिखा जाए, तो वह Programming Language उस String को Default रूप से Empty String मान लेता है। लेकिन Python में हम ये

उम्मीद नहीं कर सकते कि Python किसी String को स्वयं अपने स्तर पर Empty String मान लेगा बल्कि जब भी कभी जरूरत होगी, हमें Manually Empty String को Specify व Compare करना होगा।

इसलिए यदि आप C/C++, Java, C# जैसी Programming Languages सीखने के बाद Python सीख रहे हैं, तो तरह-तरह के Bugs से बचने का एक ही तरीका है कि आप इन Languages के Implicit तरीकों को भूल जाएं और हमेशा Explicit तरीकों पर ही विश्वास करें।

Python Codes की **Uniformity** (एकरूपता) के कारण इसमें लिखे Codes आसानी से समझ में आ जाते हैं, फिर भले ही उन Codes को आपने लिखा हो या किसी दूसरे Programmer ने। इसके अलावा Python एक ऐसा Scripting Language है जो **Functional Programming** व **Object Oriented** जैसे कई उन तरीकों को Support करता है, जिनके माध्यम से Code Reusability की सुविधा को भी प्राप्त किया जाता है। जिसके परिणामस्वरूप Python एक बहुत ही Powerful Scripting Language बन जाता है और किसी भी तरह की Programming से सम्बंधित समस्या का समाधान प्राप्त करने के लिए Python को उतनी ही सक्षमता के साथ Use किया जा सकता है जितना किसी अन्य Modern Programming Language जैसे कि C/C++, Java, C# आदि को।

Python is Developer Friendly Scripting Language

C/C++, Java, C# जैसी **Compiled** व **Statically Typed** Programming Languages की तुलना में Python Programmer की Productivity कई गुना ज्यादा Increase हो जाती है। क्योंकि इन Programming Languages द्वारा जिन कामों को करने के लिए जितने Program Codes लिखने पड़ते हैं, उनकी तुलना में Python में केवल 20% से 30% Program Codes द्वारा ही उन कामों को कर लिया जाता है।

यानी Python में हमें काफी कम Program Codes लिखने पड़ते हैं जिसकी वजह से एक Developer के रूप में हमारा काम काफी आसान हो जाता है। क्योंकि हमें कम Codes **Type** करने पड़ते हैं, कम Codes **Debug** करने पड़ते हैं और एक बार Application Complete हो जाने के बाद कम Codes **Maintain** करने पड़ते हैं। परिणामस्वरूप एक Python Programmer का जीवन, C/C++, Java, C# Programmer की तुलना में काफी आसान हो जाता है।

इतना ही नहीं, Python एक Interpreter Based Scripting Language है, इसलिए इसे Develop करने के बाद हमें इसे C/C++, Java, C# जैसी Programming Languages की तरह लम्बे Compile and Linking Process को भी Follow नहीं करना पड़ता बल्कि हम जैसे ही Program लिखते हैं, उसे तुरन्त Interpret करके उसे Output को Test कर सकते हैं और Application Development के दौरान इस बार-बार के Time Consuming व Lengthy Compiling and Linking Process से बच जाते हैं, जिससे एक Python Programmer के रूप में हमारा काम काफी आसान हो जाता है। जबकि एक बड़े C/C++, Java, C# आधारित Application Software को Development के दौरान बार-बार Compile and Linking Process Follow करना पड़ता है, जिसमें काफी ज्यादा समय बर्बाद होता है।

Python एक ऐसी Scripting Language है जिसमें कम Codes द्वारा ज्यादा Tasks Complete किए जाते हैं और इसे मूल रूप से Development की Speed को कम से कम करने के लिहाज से ही Develop किया गया है ताकि कम से कम समय में Best Possible Application Software Develop किया जा सके।

Python द्वारा Provide किए जाने वाले Simple Coding Syntax, Dynamic Typing, Compiling and Linking से सम्बंधित Lengthy Steps से छुटकारा व Built-In Toolsets की वजह से एक Python Programmer किसी भी अन्य Programming Language की तुलना में कहीं कम समय में Up and Running Application Software Develop कर पाता है जिससे न केवल Development Cost कम हो जाती है बल्कि Application को Debug, Manage व Maintain करना भी काफी आसान हो जाता है।

Python is Portable Scripting Language

लगभग सभी Python Programs बिना किसी तरह का कोई Change किए हुए किसी भी Computer System पर ज्यों का त्यों Run हो जाते हैं। यदि हमने किसी Python Program को Linux पर Develop किया हो, तो उसे MacOS या Windows System पर Run करने के लिए हमें केवल उस Program की Script को Source Computer System से Target Computer System पर Simple Text File के रूप में Copy-Paste करना होता है और अगर Target Computer System पर भी Compatible Python Interpreter Installed है, तो हम अपनी Script को तुरन्त Run कर सकते हैं।

इसके अलावा Python हमें कई Built-In तरीके Provide करता है, जिनके माध्यम से Source Computer System के GUI, Database Functionality, Web System आदि को बिना कोई Change किए हुए पूरी Compatibility के साथ Target System पर Port करके बिना किसी परेशानी के Run कर सकते हैं।

जबकि विभिन्न Operating System व Computer System Architecture पर GUI, Database, Web Systems आदि पूरी तरह से भिन्न होते हैं और Python, विभिन्न Operating System व Computer System Architecture के बीच Program Portability की सुविधा Built-In तरीके से ही Provide कर देता है।

Python have Large Collection of Supports

Python के साथ हमें **Standard Libraries** के रूप में **Prebuilt Portable Functionality** वाले **Codes** का एक विशाल Collection मिलता है जिनका प्रयोग करके हम विभिन्न प्रकार की Common जरूरतों को बहुत ही आसानी से पूरा कर सकते हैं। ये Libraries विभिन्न प्रकार के Application Level Programming Tasks, Pattern Matching, Network Scripting जैसी जरूरतों को आसानी से पूरा कर सकते हैं।

Python का Standard Libraries का ही विशाल Collection नहीं है, बल्कि विभिन्न प्रकार की जरूरतों को पूरा करने के लिए हमें ढेर सारी Third-Party Libraries भी मिलती हैं, जो उन जरूरतों को बड़ी ही आसानी से पूरा करने में उपयोगी साबित होती हैं, जिन्हें Standard Libraries द्वारा आसानी से पूरा नहीं किया जा सकता।

इतना ही नहीं, Python हमें ये सुविधा भी देता है कि हम किसी भी Standard Library अथवा Third-Party Library को Extra-Ordinary जरूरतों को पूरा करने के लिए बड़ी ही आसानी से Extend भी कर सकते हैं।

Python की **Flask, Django** जैसी Third-Party Libraries का प्रयोग करके हम बड़ी ही आसानी से Websites व Web Applications Develop कर सकते हैं। जबकि *SciPy, NumPy, scikit-learn, TensorFlow* जैसी Third-Party Libraries का प्रयोग करके हम Python को **Artificial Intelligence** व **Machine Learning** जैसी जरूरतों को पूरा करने के लिए Use कर सकते हैं।

इन विभिन्न प्रकार की Libraries को Support करने के कारण ही Python को Multi-Purpose Scripting Language कहा जाता है क्योंकि विभिन्न प्रकार की Supporting Libraries का प्रयोग करके हम एक Python Scripting Language द्वारा कई तरह के Complex Tasks को भी Successfully Complete कर लेते हैं।

Python Supports Multiple Types of Component Integration

Python Scripts विभिन्न प्रकार के Integration Mechanisms को Use करके Application के विभिन्न Parts से आसानी से Communicate कर सकता है। Python Script को हम एक *Customization* व *Extension* Tool के रूप में भी Use कर सकते हैं। Python द्वारा हम C/C++ Code Libraries को Invoke कर सकते हैं और C/C++ Programs द्वारा Python Script को Call कर सकते हैं।

इसी तरह से Python के साथ Java व .NET Components को भी Integrate कर सकते हैं और Serial Port पर COM Framework के माध्यम से SOAP, XML-RPC, CORBA जैसे Interfaces द्वारा किसी दूसरी Device के साथ Interact भी कर सकते हैं।

Disadvantages of Python

हालांकि कोई भी Programming या Scripting Language Perfect नहीं है और Python भी उन्हीं में से एक है लेकिन Python का केवल एक ही Disadvantage है और वो है इसकी *Performance* यानी *Execution Speed* जो कि हमेशा उतनी Fast नहीं होती जितनी C/C++ जैसी Compiled Languages की होती है। इसलिए कभी-कभी ऐसी परिस्थितियां पैदा हो जाती हैं जहां High Performance इतना

ज्यादा Critical Role Play करता है कि हमें उन Specific Tasks को Accomplish करने के लिए C/C++ जैसी **Lower Level Languages** को Use करना पड़ता है।

क्योंकि C/C++, Underlying Hardware को Directly Access and Manipulate करने में सक्षम है क्योंकि ये Compiled Language है इसलिए इसके Programs, Underlying *Operating System* व *Machine Architecture* के अनुरूप **Directly Executable Binary Codes** में Translate हो जाते हैं, जबकि Python, कभी भी Underlying Hardware को Directly Access नहीं कर सकता क्योंकि जब भी Python Script को Run करते हैं, तो वह Java व C# की तरह एक Intermediate Executable Byte Code में Translate होता है और फिर इस Bytecode को **Python Interpreter** द्वारा Executable Binary Codes में Translate करके Execute किया जाता है।

इसलिए जिस तरह से C/C++ Executable Codes, Directly Underlying Hardware Architecture के साथ Mapped रहते हैं, ठीक उसी तरह से Python के Bytecodes, Underlying Hardware Architecture के साथ Directly Mapped नहीं रहते, बल्कि Execute होते समय इन्हें Python Interpreter की एक और Layer से गुजरना पड़ता है। परिणामस्वरूप Python Programs की Execution Speed हमेशा ही C/C++ Program की तुलना में कुछ कम होना स्वाभाविक है।

हालांकि Python की Performance को प्रभावित करने वाली इसी समस्या का एक फायदा भी है जो कि इसे एक पूरी तरह से Portable Scripting Language बना देता है। जिसकी वजह से हम किसी भी Operating System व Underlying Hardware Architecture पर Create किए गए Application को बड़ी ही आसानी से बिना कोई परिवर्तन किए हुए ज्यों का त्यों किसी दूसरे Operating System व Underlying Hardware Architecture पर Run कर सकते हैं।

यद्यपि Python के Codes Bytecodes में Translate होते हैं, फिर भी PyPy जैसे कुछ Systems के माध्यम से हम Python Codes को भी Directly Machine Executable Codes में Compile कर सकते हैं जिसके परिणामस्वरूप Python Codes की Execution Speed 10 से 100 गुना तक बढ़ जाती है।

जब हम C/C++ जैसे Low Level Compiled Machine Codes के साथ Python Codes के Execution Speed की तुलना करते हैं, तब निश्चित रूप से Python Codes की Execution Speed कम होती है लेकिन फिर भी समय-समय पर Python Interpreter को इस तरह से Optimize किया जाता रहा है, जिससे Python की Performance भी किसी भी अन्य Scripting Language की तुलना में काफी अच्छी है।

यहां तक कि जब हम हमारे Python Script में GUI Implement करते हैं या File Processing करते हैं, तो उस समय हमारे Python Script की Speed उतनी ही होती है, जितनी C Language के Compiled Codes की होती है क्योंकि GUI या File Processing से सम्बंधित Tasks को Perform करने के लिए वास्तव में C की Compiled Codes ही Execute होते हैं। यानी Python, इस तरह के Tasks से सम्बंधित

Script Codes को Python Interpreter के अन्दर Exist उस हिस्से पर Dispatch कर देता है, जो Compiled C Codes को Execute करने का काम करता है।

इसलिए भले ही Python Bytecodes की Speed, C/C++ Compiled Codes की तुलना में कम हो, लेकिन Python हमें ये सुविधा भी देता है कि हमें जिन Program Codes के लिए Compiled Codes जितनी Execution Speed की जरूरत हो, उन्हें अलग करके C Language के Compiled Codes द्वारा Process करवा सकते हैं और फिर Return होने वाले Result को फिर से Python Script में Retrieve कर सकते हैं। क्योंकि Python, किसी भी C/C++ Compiled Codes को बड़ी ही आसानी से Invoke कर सकता है।

Python Popularity

Python वर्तमान समय की विभिन्न Modern Programming Languages में सबसे पुरानी और सबसे Popular Language है और इसका एक मुख्य कारण ये भी है कि Python एक **Open Source** Language है इसलिए जो लोग Language License के रूप में सालाना खर्चा करना नहीं चाहते, वे लोग Python को ज्यादा Priority देते हैं।

क्योंकि ये काफी पुरानी Scripting Language होने की वजह से लगभग सभी तरह की जरूरतों को पूरा करने से सम्बंधित Python Library आसानी से उपलब्ध हैं साथ ही उन Libraries को Use करने और उनसे सम्बंधित Errors को Debug करने से सम्बंधित पर्याप्त Documentation, Free Forum व Active Developer Community भी पहले से उपलब्ध है। इसलिए Python से सम्बंधित किसी भी तरह की समस्या के सामाधान के लिए हमें किसी Specialized License की जरूरत नहीं है।

Linux Distributions के साथ Python पहले से Installed रहता है और वर्तमान समय में भी लगभग 70% Web Server Linux Distribution पर ही आधारित हैं। इसलिए Linux Distributions व MacOS के लिए Python एक Default Language की तरह Use होता है। साथ ही इतनी ज्यादा पुरानी होने के कारण Python अब काफी Stable व Robust Scripting Language बन चुकी है। इसलिए बिना किसी दिक्कत के हम इसे छोटे या बहुत बड़े, किसी भी तरह के Application को Develop करने के लिए पूरी सफलता के साथ Use कर सकते हैं क्योंकि बहुत बड़ी-बड़ी Companies भी इसे अपने ज्यादातर Projects में Use करती हैं। उदाहरण के लिए-

- ✓ **NASA** अपने Research व Scientific Programming से सम्बंधित जरूरतों को पूरा करने के लिए Python को ही मुख्य Scripting Language के रूप में Use करता है।
- ✓ **Raspberry Pi**, जो कि एक *Single Board Computer System* है, Python को मुख्य Educational Language के रूप में Use करता है।
- ✓ **BitTorrent** जो कि एक *Peer-to-Peer File Sharing System* है, उसे भी Python Language में ही Develop किया गया है।

- ✓ कई तरह की Animated Movies (*Pixar, Industrial Light & Matic, etc...*) को भी Python का प्रयोग करते हुए ही बनाया गया है।
- ✓ **Google** अपने *Search System* में Python का बहुत उपयोग करता है।
- ✓ **YouTube** *Video Sharing Service* को मूलतः Python में ही लिखा गया है।
- ✓ **Dropbox** *Storage Service* के Client व Server दोनों Applications को मुख्यतः Python में ही
- ✓ Google का **App Engine** *Web Development Framework* में भी Python को एक Language के रूप में Use किया जाता है।
- ✓ **Intel, Cisco, HP, Seagate, Qualcomm** व **IBM**, *Hardware Testing* के लिए Python को ही मुख्य Language के रूप में Use करते हैं।
- ✓ **JPMorgan** जैसे Financial Institutions, अपने Financial Market से सम्बंधित Forecasting के लिए Develop किए गए Algorithms को Python के माध्यम से ही Develop व Control करती हैं।
- ✓ **Maya** नाम का एक Powerful *Integrated 3D Modeling and Animation System* है, जो कि Python Language को एक Scripting API के रूप में Support करता है।
- ✓ Python का प्रयोग वर्तमान समय में ढेर सारी बड़ी कम्पनियां **Robotics** (*Artificial Intelligence* व *Machine Learning*) के लिए Use कर रही हैं।
- ✓ **Netflix** व **Yelp** में भी Backend को Python द्वारा ही Successfully Handle व Control किया जाता है।

इस तरह से हम समझ सकते हैं कि Python कोई ऐसी-वैसी Scripting Language नहीं है बल्कि इतनी Powerful है कि बड़ी-बड़ी कम्पनियां अपने Internal Projects के लिए इसे मुख्य Language के रूप में Use करती हैं और Python किसी एक तरह के काम को Best तरीके से पूरा करता हो, ऐसा नहीं है बल्कि Python को किसी भी तरह की Programming जरूरत को पूरा करने के लिए और किसी भी Domain से सम्बंधित Application Develop करने के लिए पूरी सक्षमता व विश्वास के साथ Use किया जा सकता है।

इसीलिए Python को Multi-Purpose या General-Purpose Scripting Language कहा जाता है, जिसमें हम छोटे-मोटे Short-Term Tasks Complete करने से लेकर बड़े-बड़े Long-Term Application Projects तक किसी भी तरह के Software Develop, Manage, Extend, Administer, Maintain, व Deploy कर सकते हैं।

Where is Used Python?

Python एक Well Designed Programming Language है और दुनियाँ के किसी भी तरह के Real World Computing Task को Accomplish करने में सक्षम है। इसे किसी एक तरह की जरूरत को पूरा करने के लिए नहीं Develop किया गया है, बल्कि हम इसे किसी भी तरह के Domain से सम्बंधित Application Develop करने के लिए Use कर सकते हैं।

परिणामस्वरूप एक General-Purpose Scripting Language होने की वजह से Python को असीमित तरीकों से Use करते हुए अनन्त प्रकार के Applications Develop किए जा सकते हैं। हम इसे Web Application Develop करने के लिए भी उतने ही Powerful तरीके से Use कर सकते हैं, जितना **Game** Develop करने, **Robotics** के *Artificial Intelligence and Machine Learning* Algorithms को Develop करने अथवा **Spacecraft** के *Operating System* Program को Develop करने अथवा *Operating System* Program को Network के माध्यम से *Remotely Control* करने के लिए कर सकते हैं।

हालांकि हम Python का प्रयोग करते हुए किसी भी तरह की जरूरत को पूरा कर सकते हैं, लेकिन फिर भी Python की कुछ बहुत उपयोग होने वाली Capabilities को अग्रानुसार समझा जा सकता है-

System Programming

Python में Underlying Operating System को Access करने से सम्बंधित Built-In Interfaces हैं, जिसकी वजह से Maintainable System-Administration Tools व Utilities जिन्हें Shell Tools भी कहते हैं, Develop करना काफी आसान व सुविधाजनक हो जाता है।

उदाहरण के लिए हम किसी भी Python Program में इन Built-In Interfaces के माध्यम से हम बड़ी ही आसानी से Underlying Operating System के File System में किसी भी *File and Directory Trees* की Searching कर सकते हैं, अन्य Executable Programs को Launch कर सकते हैं, Processes व Threads के साथ Parallel Processing कर सकते हैं और ऐसे ही कई और तरह के Task Accomplish कर सकते हैं, जिन्हें Perform करने के लिए अन्य Programming Languages में काफी लम्बा Procedure Follow करना पड़ सकता है।

हम Python में ऐसा इसलिए कर सकते हैं क्योंकि Python Interpreter, C/C++ जैसी Lower Level Languages को काफी Integrated तरीके से Use करता है और लगभग सभी Operating Systems अपने Core Level पर C/C++ व Assembly Language Codes पर ही निर्भर हैं क्योंकि Operating System के Core Level पर High Performance की जरूरत होती है, जो कि केवल C/C++ व Assembly Language के माध्यम से ही सम्भव है और Python इन Lower Level Languages के साथ काफी बेहतर तरीके से Integrated है।

Python की **Standard Library** *POSIX Bindings* के साथ उपलब्ध है जो कि लगभग सभी तरह के **Usual OS Tools** जैसे कि *Environment Variables, Files, Sockets, Pipes, Processes, Multiple Threads, Regular Expression Pattern Matching, Command Line Arguments, Standard Stream Interfaces, Shell-Command Launchers, Filename Expansion, Zip File Utilities, XML व JSON Parsers, CSV File handlers* आदि और भी ढेर सारी तरह की Functionalities को Core Level पर Support करता है।

यानी इन Operating System Level की जरूरतों को पूरा करने के लिए हमें अलग से किसी Library को Use नहीं करना पड़ता। Python की Standard Library इन सभी OS Level Tools and Utilities को Default रूप से Support करता है।

इतना ही नहीं, Python के System Interfaces को पूरी तरह से Portability Support के साथ Design किया गया है। ताकि एक ही Standard Library विभिन्न प्रकार के Operating System व Underlying Hardware को समान रूप से Access व Manipulate कर सके।

उदाहरण के लिए Python की जो Script, Linux OS के लिए Directory Trees को Copy करता है, Windows OS के लिए भी वही Script समान प्रकार से Directory Tree को Copy कर लेता है। जबकि दोनों ही Operating Systems पूर्णतः भिन्न तरीके से काम करते हैं और ऐसा इसलिए होता है क्योंकि Python Interpreter की Standard Library को पता रहता है कि वह किस Operating System व Underlying Hardware Architecture पर Installed है जिसके आधार पर वह इस बात का निर्णय ले पाता है कि Linux OS के Directory Structure को किस तरह से Copy करना है और Windows OS के Directory Structure को किस तरह से Copy करना है।

Graphical User Interfaces – GUIs

जब हम Python को **Desktop Applications** Develop करने के लिए Use करते हैं, तब Python हमें कई अलग तरह की GUI Libraries Provide करता है, जिनकी वजह से Python का GUI Designing Process काफी Simple व Fast हो जाता है। Python में GUI Develop करना इतना आसान होता है कि एक Simple Window Create करने के लिए हमें केवल 3 से 4 Lines का Python Code लिखना पड़ता है, जबकि इसी काम को यदि हम Standard C Library के माध्यम से करें, तो हमें 100 से ज्यादा Lines के Codes लिखने पड़ सकते हैं।

Python Default रूप से Desktop GUI Develop करने के लिए **Tk GUI API** के एक *Standard Object Oriented Interface* को Support करता है जिसे **tkinter** के नाम से जाना जाता है। tkinter हमें Portable Native GUI Develop करने की सुविधा देता है। यानी जब हम tkinter का प्रयोग करते हुए GUI Design करते हैं, तब हमारे GUI का Look and Feel पूरी तरह से हमारे Operating System के अनुसार होता है इसीलिए Current Operating System के सभी अन्य GUI का जो Standard Look and Feel होता है, हमारे Python Program के GUI का Look and Feel भी उसी तरह का होता है।

जब हम Python/tkinter GUI Develop करते हैं, तो हमारा GUI **Windows, X-Windows** (*for Unix and Linux*), **MacOS** (*on Classic and OS X both*) सभी पर बिना किसी Change के एक समान Run होता है।

tkinter के साथ ही यदि हम **PMW** नाम के एक *Free Extension Package* को भी Use कर लें, तो हमारे tkinter GUI Toolkit में कुछ Advanced Widgets और Add हो जाते हैं, जो कि tkinter की

Standard Library में उपलब्ध नहीं होते। इन Special Widgets का प्रयोग हम कुछ अलग तरह की जरूरतों को पूरा करने के लिए कर सकते हैं जो कि tkinter की Standard Library में उपलब्ध नहीं हैं। उदाहरण के लिए *Notebooks, Comboboxes, Selection, Paned, Scrolled, Dialog Windows, etc...* Standard tkinter Library में उपलब्ध नहीं हैं, लेकिन इन्हें PMW से प्राप्त किया जा सकता है।

इसके अलावा **C++ Library** पर आधारित **wxPython GUI API** का प्रयोग करके भी हम Python GUI Design कर सकते हैं जो कि हमें Python में Portable GUI Construct करने की सुविधा देने वाला Alternative Toolkit है।

Dabo नाम का एक Higher Level GUI Development Toolkit भी है जिसे **wxPython** व **tkinter** की Base API के आधार पर Develop किया गया है।

इनके अलावा और भी कई तरह की Libraries हैं, जिनका प्रयोग करके हम High Quality के Python GUI Develop कर सकते हैं। उदाहरण के लिए *PyQt, GTK with PyGTK, MFC with PyWin32, .Net with IronPython, Jpype* और *Swing with Jython* जो कि Python का एक Java Version है, इन सभी के माध्यम से भी हम Python GUI Design कर सकते हैं।

जो Applications, Web Browser में Run होते हैं, या जिनके लिए Simple User Interface की Requirement होती है, उनके लिए हम **Jython, Python Web Frameworks** (*Flask, Django, etc.*) अथवा **Server-Side CGI Scripts** को भी Use कर सकते हैं, जिसमें User Interface को सामान्यतः HTML5/CSS3 के माध्यम से बनाया जाता है क्योंकि Web Browser में केवल इन्हीं के माध्यम से GUI बनाया जा सकता है।

Internet and Network Programming

Python में **Standard Internet Modules** को भी Standard Library के रूप में Included रखा गया है जो कि हमें Client Side व Server Side से सम्बंधित विभिन्न प्रकार के Networking Tasks को आसानी से Accomplish करने की सुविधा देते हैं। उदाहरण के लिए-

- ✓ Client-Server के बीच **Sockets** के माध्यम से Communicate कर सकते हैं।
- ✓ **Server Side CGI Scripts** पर Send किए गए HTML Form की Information को Extract कर सकते हैं।
- ✓ **FTP** के माध्यम से Network पर File Transfer कर सकते हैं।
- ✓ **XML** व **JSON** Documents को Generate व Parse कर सकते हैं।
- ✓ **Emails** को *Send, Receive, Compose* व *Parse* कर सकते हैं।
- ✓ **URLs** द्वारा Web Pages को Fetch कर सकते हैं।
- ✓ Client-Server के बीच **XML-RPC, SOAP** व **Telnet** के माध्यम से Communication कर सकते हैं।

इसी तरह के और भी कई Tasks हैं, जिन्हें Accomplish करने के लिए हमें अलग से किसी Library की जरूरत नहीं पड़ती। Python विभिन्न प्रकार की Basic Networking से सम्बंधित जरूरतों को पूरा करने के लिए सभी जरूरी Codes, Core Libraries या Standard Libraries के रूप में ही Provide कर देता है।

इसके अलावा Web पर Third-Party Tools के रूप में Internet Programming करने से सम्बंधित Libraries का Large Collection उपलब्ध है, जिन्हें Use करने के लिए एक Programmer को कुछ भी Pay नहीं करना होता और Programmer के रूप में हम इन Free Third-Party Libraries का प्रयोग करके Internet व Web Programming से सम्बंधित विभिन्न प्रकार की जरूरतों को आसानी से पूरा कर सकते हैं। उदाहरण के लिए-

- 1 **HTMLGen System**, Python की Class Based Description के आधार पर HTML Files Generate कर देता है।
- 2 **mod_python Package** के माध्यम से Apache Web Server के अन्दर Python काफी Efficiently Run हो जाता है और **Python Server Pages** के माध्यम से Server-Side Templating को भी Support करता है।
- 3 **Jython** System, Python/Java Integration की सुविधा Provide कर देता है जिसके द्वारा हम Java Programming Language का प्रयोग करते हुए ऐसे Server Side Applets (**Servlets**) Code कर सकते हैं जो कि Client Side में Run होता है।
- 4 **Django, Flask, TurboGears, WebWare, Pylons** व **Zope** जैसे Full-Fledged Web Development Frameworks का प्रयोग करके हम Python पर आधारित किसी भी तरह के Web Application या Web Site/Blog को बड़ी ही आसानी से Develop कर सकते हैं। ये Third-Party Tools हमें Quickly Production Ready Web Applications Develop करने में सहायक होते हैं, जिनमें **Object-Relational Mappers (ORM)**, **Model-View-Controller (MVC)** Architecture, Server-Side Scripting and Templating व AJAX Support जैसे Features होते हैं जो हमें Enterprise स्तर के सम्पूर्ण Web Development Solutions उपलब्ध करवाते हैं।
- 5 अब Python के लिए हमें ऐसे Packages भी मिलते हैं जो **Rich Internet Applications (RIA)** की सुविधा Provide करते हैं जहां हम Python-to-JavaScript Compiler, AJAX Framework and Widget Set, जैसे Tools का प्रयोग करते हुए विशिष्ट प्रकार के Rich Web Applications Develop कर सकते हैं।
- 6 Python को अब **App Engine** द्वारा Cloud Computing में भी बहुत Use किया जा सकता है।

Component Integration

Python इतनी Flexible Scripting Language है कि हम बड़ी ही आसानी से इसमें C/C++ के Program Codes Embed कर सकते हैं और Special प्रकार की जरूरतों को C/C++ Codes द्वारा पूरा कर सकते हैं जिसे Python Scripting द्वारा Directly Implement करना काफी जटिल हो सकता है।

उदाहरण के लिए हम अपने Python Script में किसी C Library को Integrate कर सकते हैं जो Python को उस C Library के Components को Launch व Test करने के लिए Enable कर देता है। इसी तरह से हम किसी Onsite Project के Customization के लिए Python Script को उसमें Embed कर सकते हैं और बिना उस सम्पूर्ण Project को Recompile किए हुए उसे Python Script के माध्यम से Customize कर सकते हैं।

SWIG व SIP Code Generators जैसे Tools, Compiled Components को Python Script में Use करने के लिए जरूरी सभी Tasks को Automatically Fulfill कर देते हैं। जबकि Cython System, Programmer को Python व C/C++ जैसे Codes को आपस में Mix करके Use करने की सुविधा देता है।

Python का **COM Support**, Java-based Implementation **Jython** और .NET-based Implementation **IronPython** द्वारा हमें Windows Components को Script करने का एक Alternate तरीका Provide करता है। यानी Python Scripts ऐसे Frameworks Use कर सकता है जिनके माध्यम से हम Windows Platform के MS-Word, MS-Excel आदि को Script कर सकते हैं, जो कि Python के COM Support की वजह से ही सम्भव है।

Database Programming

वर्तमान समय में जितने भी जितने भी Commonly Use होने वाले Relational Database Systems जैसे कि **Oracle, MySQL, MSSQL Server, Sybase, Informix, Oracle, ODBC, PostgreSQL, SQLite** आदि हैं, उन सभी के लिए Python Interface उपलब्ध हैं। यानी हम किसी भी प्रकार के RDBMS के साथ Python को Connect कर सकते हैं और उसमें Data Store, Access व Manipulate कर सकते हैं।

Python में Python Scripts से SQL Database Systems को Access करने के लिए एक **Portable Database API** भी Define किया गया है जो कि सभी प्रकार के Underlying Database को समान रूप से समान Code द्वारा Access and Manipulate करने की सुविधा Provide करता है।

उदाहरण के लिए यदि हम हमारे Application के लिए Backend Database के रूप में Oracle Use कर रहे हैं, लेकिन कुछ समय बाद हम अपने Underlying Backend Database को Oracle से Change करके MySQL पर Shift करना चाहते हैं, तो इस Portable Database API को Use करने की स्थिति में

हमें Underlying Backend Database को Change करने के लिए केवल अपने Python Program में Database Connectivity से सम्बंधित केवल एक Line के Program Code को Modify करते हुए Underlying Database System के **Name, Database** व **Credentials** को Change करना होता है और बाकी के सभी जरूरी Changes को वह API स्वयं अपने स्तर पर Handle कर लेता है। जबकि यदि हम इस API को Use न करें, तो उस स्थिति में Underlying Database System को Change करना एक Complex, Time Consuming व Buggy प्रक्रिया बन जाती है।

Default रूप से Python 2.5 से ही In-Process SQLite Embedded SQL Database Engine को Standard Part के रूप में Include कर लिया गया है जो कि Prototyping व Basic Program Storage दोनों तरह की जरूरतों को पूरा करने के लिए उपयोगी है। इसलिए छोटी-मोटी जरूरतों को पूरा करने के लिए जब हमें Database की जरूरत होती है, तब हम इस Embedded SQLite Database को सफलतापूर्वक उपयोग में ले सकते हैं। हालांकि जब हमें बड़े Applications Develop करने होते हैं, जहां Underlying Database हमारे Application का एक बहुत ही महत्वपूर्ण Part होता है, तब हम इस Embedded SQLite Database पर पूरी तरह से निर्भर नहीं हो सकते।

जब हम Non-SQL आधारित तरीके से अपने Application के Data को Handle करना चाहते हैं, तब Python का **pickle** Module हमें एक **Simple Object Persistence System** Provide करता है, जिसका प्रयोग करके हम आसानी से Entire Python Object को किसी File या File जैसे Object में Save कर सकते हैं व बाद में जरूरत पड़ने पर फिर से उस File / File Object से Restore कर सकते हैं।

Web पर हमें **ZODB** व **DURUS** नाम के Third-Party Open Source Systems मिलते हैं, जो हमें Python Script के लिए Complete Object Oriented Database System Provide करते हैं।

इसी तरह से **SQLObject** व **SQLAlchemy** नाम की दो Third-Party Libraries हैं, जो Relational Object Mappers (**ORM**) Implement करते हैं और Python के **Model** Classes को Underlying Database की **Relational Tables** के साथ Map कर देते हैं।

PyMongo Interface द्वारा हम MongoDB Database के साथ Interact करने की सुविधा प्राप्त करते हैं। MongoDB एक High-Performance, Non-SQL, Open-Source JSON Style का **Document Database** है जो कि Data को Python के List व Dictionary Data Structure के Format में Key-Value Pair के रूप में Store करता है। इसीलिए इसके Data को Python के Standard Library में उपलब्ध **json Module** द्वारा Parse करके आसानी से Access and Manipulate किया जा सकता है।

इसके अलावा Google के **App Engine** की तरह Data को Python का प्रयोग करते हुए Store करने के कुछ और Specialized तरीके भी हैं जो कि Data को Python Classes के साथ Model करता है और Extensive Scalability Provide करता है। इसके अलावा **Azure, PiCloud, OpenStack** व **Stackato** जैसे कुछ और Emerging Cloud Storage Options भी हैं, जिनका प्रयोग Python के माध्यम से Data को Store, Access व Manipulate करने के लिए किया जा सकता है।

Rapid Prototyping

Python में लिखे गए Programs व Components काफी हद तक C Program की तरह ही दिखाई देते हैं। जिसकी वजह से किसी नए System का Prototype बनाने के लिए Initially Python को Use करना ज्यादा आसान व सुविधाजनक होता है और जब एक बार Python Components का प्रयोग करते हुए Prototype बना लिया जाता है, उसके बाद उस Prototype को बनाने के लिए Use किए गए Selected Components को C/C++ जैसी Compiled Languages द्वारा Compile कर लिया जाता है और Final System को Deliver कर दिया जाता है।

कुछ अन्य Prototyping Tools की तरह Python में Create किए गए Prototype को Complete होने के बाद, फिर से पूरी तरह से C/C++ जैसी Compiled Language में Translate करना जरूरी नहीं होता। बल्कि केवल उन Components को ही Compile करना होता है, जिन्हें Compile करने के लिए Select किया गया है और Python Prototype के जिन हिस्सों या Components को Efficiency के लिए Compile करना जरूरी नहीं होता, उन्हें Compile नहीं किया जाता। जिसके कारण Python का प्रयोग करते हुए Develop किए गए Prototype को Manage, Extend, Maintain व Deploy करना ज्यादा आसान रहता है।

Scientific and Numeric Programming

Numeric Programming एक ऐसा क्षेत्र है, जहां सामान्यतः किसी न किसी Compiled Programming Language को Main Programming Language के रूप में प्राथमिकता देते हैं क्योंकि Numerical Programming में ढेर सारी Calculations होने की वजह से Code Efficiency व High Performance की अत्यधिक जरूरत रहती है।

लेकिन Scripting Language होने के बावजूद अब Python इतना सक्षम है कि किसी भी अन्य Compiled Programming Language के समान ही इसका प्रयोग भी किसी भी तरह की **Numeric Programming** के लिए उन्ती ही सफलता के साथ किया जा सकता है, जितना C/C++ जैसी Modern High Level Compiled Programming Language को और इन्हीं के समकक्ष Code Efficiency व High Performance को भी प्राप्त किया जा सकता है।

NumPy, Python के लिए Develop किया गया ऐसा ही एक High Performance Numeric Programming Extension है। Python को High Speed Performance प्राप्त करने के लिए Compiled Languages Numeric Routines Codes के साथ Integrate करके NumPy, Python को एक Easy to Use Numeric Programming Tool बना देता है जो कि अक्सर FORTRAN या C++ जैसी Compiled Language में Developed Codes की Performance के लगभग समान Performance देने में सक्षम हो जाता है।

हम जब भी कभी Python का प्रयोग करते हुए किसी प्रकार का **Animation, 3D-Visualization, Parallel Processing** या **Graphics** से सम्बंधित Application Develop करते हैं, तब हमें कई तरह के **Numeric Tools** Use करने की जरूरत पड़ती है क्योंकि इस तरह के Applications में Internally ढेर सारी Mathematical Calculations Perform होती हैं। इस तरह के Applications Develop करने के लिए जरूरी Tools हमें Python के **SciPy** व **ScientificPython** Extension से प्राप्त होते हैं।

उदाहरण के लिए हम **NumPy** को Core Component की तरह Use कर सकते हैं और अपने Application की जरूरत के अनुसार **NumPy** के साथ **SciPy** व **ScientificPython** Extensions को Use करके Additional प्रकार की जरूरतों से सम्बंधित Features प्राप्त कर सकते हैं।

NumPy के अलावा Python के **PyPy Implementation** को Use करके भी हम Numeric Programming कर सकते हैं और इस तरह के Scientific व Numerical Calculation Heavy Applications Develop कर सकते हैं, जो कि NumPy की तुलना में 10 से 100 गुना ज्यादा तेज गति से Numerical Calculations Perform करता है।

Designing, Image Processing and Game Programming

Python का प्रयोग करते हुए हम Games भी Develop कर सकते हैं। जब हम Python का प्रयोग करते हुए Game Programming करना चाहते हैं, तब हम **pygame, cgkit, piglet, PySoy, Panda3D** जैसे Python Extensions को Use कर सकते हैं।

इसी तरह से **PIL** व इसके नए Extension **Pillow, PyOpenGL, Blender** व **Maya** का प्रयोग करते हुए Image Processing से सम्बंधित Applications Develop कर सकते हैं।

Media File Content व Metadata Tag Processing के लिए हम Python के **PyMedia, ID3, PIL/Pillow** Extensions का प्रयोग कर सकते हैं।

Hardware, System and Embedded Programming

Python का प्रयोग करते हुए हम **pySerial Extension** का प्रयोग करके Window व Linux के लिए Serial Port Communication से सम्बंधित Device Driver Programming कर सकते हैं।

इसी तरह से **PyRo** Toolkit का प्रयोग करते हुए हम Robot Control Programming कर सकते हैं। जबकि Artificial Intelligence व Machine Learning से सम्बंधित Applications Develop करते समय Natural Language Analysis करने के लिए हम **NLTK** Package का प्रयोग कर सकते हैं।

Raspberry Pi व **Arduino** Boards की Programming के लिए भी हम Python को सफलतापूर्वक Use करते हुए Embedded Programming कर सकते हैं।

Artificial Intelligence के लिए हम **PyBrain** Nural Net Library और **Machine Learning** के लिए हम Python के साथ **Milk** Library को Use कर सकते हैं।

Expert System Programming के लिए **PyCLIPS**, **Pyke**, **Pyrolog** व **pyDatalog** Libraries को Use किया जाता है।

Android व **iOS** Platform पर आधारित Mobile व Tablet PC के Apps Develop करने के लिए भी हम Python को Use कर सकते हैं।

Document and Data Processing Programming

Data Processing और **Report Generation** के लिए हम **ReportLab**, **Sphinx**, **Cheetah**, **PyPDF** Use कर सकते हैं।

इसी तरह से **Data Visualization** के लिए हम **Mayavi**, **matplotlib**, **VTK**, **VPython** Use कर सकते हैं।

जबकि **json** व **csv** Modules का प्रयोग करके हम **JSON** व **CSV** File Processing कर सकते हैं।

साथ ही **DataNitro** या **PyXLL** का प्रयोग करते हुए **Excel Spreadsheet Function** व **Macro Programming** भी कर सकते हैं।

इसी तरह से Data Mining के लिए हम **Orange Framework**, **Pattern Bundle**, **Scrapy** Use कर सकते हैं, अथवा हम अपने स्वयं के Custom Codes भी लिख सकते हैं।

इन सभी के अलावा हम हमारे Day to Day Tasks को Accomplish करने के लिए अपने स्वयं के Custom Python Code Script भी लिख सकते हैं। उदाहरण के लिए System Administration करने, Emails Processing करने, अपने Documents व Media Library को Manage करने जैसे Day to Day Tasks को Complete करने के लिए हम अपने स्वयं की Custom Python Script भी लिख सकते हैं।

Python Features for Developer?

Python अपने आप में एक सम्पूर्ण Language है जो किसी भी तरह की जरूरत को पूरा करने में सक्षम है। फिर भी एक Developer के रूप में ये हमें अग्रानुसार कई ऐसे Special Features Provide करता है, जो कि अन्य Commercial Languages से प्राप्त नहीं होते।

Python is Object Oriented, Procedural and Functional

Python को शुरू से ही एक Object Oriented Language की तरह ही Develop किया गया है। इसलिए इसका **Class Model**, *Polymorphism*, *Operator Overloading* व *Multiple Inheritance* जैसे सभी Advanced OOPS Concepts को पूरी तरह से Support करता है और क्योंकि Python का Programming Syntax काफी आसान है, इसलिए OOPS Concepts को Python के अन्तर्गत Implement करना किसी भी अन्य Programming Language की तुलना में कहीं ज्यादा Easy है।

Python का शुरू से ही OOPS Nature का होना, इसे अन्य Object Oriented System Languages के लिए एक Ideal **Scripting Tool** बना देता है। उदाहरण के लिए उपयुक्त Glue Codes के माध्यम से एक Python Programs, **C++**, **Java**, **C#** जैसी किसी भी अन्य Language में Defined Class को Inherit कर सकता है।

OOPS, Python का एक ऐसा Option है, जिसके अन्तर्गत Objects को ज्यादा गहराई से जाने समझे बिना भी हम बड़ी ही आसानी से Object Oriented Programs Develop कर सकते हैं। इतना ही नहीं, C++ की तरह ही Python भी एक Hybrid Language है जिसमें हम Procedural Programming Approach को भी उतनी ही सफलता के साथ Implement कर सकते हैं जितनी सफलता के साथ Object Oriented Programming Approach को।

इस तरह से Python **Statement-Based Procedural Programming Paradigm** को भी उतने ही बेहतर तरीके से Support करता है जितना **Class-Based Object Oriented Programming Paradigm** को।

इसके अलावा Python के Recent Versions अब Built-In तरीके से **Functional Programming** को भी Procedural व OOPS के समान ही Support करने लगे हैं जिसके अन्तर्गत **Generators, Closures, Maps, Decorators, Anonymous Functions, Lambdas** व **First-Class Function Objects** मुख्य Components होते हैं, जिन्हें Python के In-Built Supported OOPS Tools के **Alternative** व **Complement** दोनों के रूप में Use किया जा सकता है।

Free and Open Source

एक Developer के लिए Python पूरी तरह से **Free** व **Open Source** है। इसलिए एक Developer न केवल Python का प्रयोग करते हुए Commercial Applications Develop कर सकता है बल्कि स्वयं Python व उसकी Core Libraries में अपने मनचाहे Changes करके उस Customized Python को जिस तरह से चाहे उस तरह से Modify, Use या Distribute कर सकता है।

Python पूरी तरह से Free होने के बावजूद उसी तरह से Supported है, जिस तरह से कोई Commercial Programming Language होती है। इसलिए Python में किसी भी तरह की समस्या आने पर आप

Python Community में उस समस्या को Discuss कर सकते हैं और Python की Community में आपको उस समस्या का उतनी ही तेज गति से समाधान भी प्राप्त होगा, जितना किसी Commercial Language की Community में प्राप्त होता है। इसलिए Python Free होने के बावजूद एक Online Community Driven Fully Supported Programming Language है।

चूंकि एक Developer के रूप में हमारे लिए Python का पूरा Source Code Accessible रहता है, इसलिए यदि किसी Specific प्रकार की जरूरत को पूरा करने में Python पूरी तरह से सक्षम न हो, तो हम हमारी उस Specific जरूरत के अनुसार Python के Source Code को Modify कर सकते हैं, जो कि केवल Open Source Programming Languages के लिए ही सम्भव है।

Python is Portable

Python का Standard Implementation मूलतः **ANSI C** Language में किया गया है जो कि वर्तमान में उपलब्ध लगभग सभी Platforms पर Compile व Run होता है। यानी Python, Mobile Device से लेकर Super Computers तक में समान रूप से Run हो सकता है, जो कि ज्यादातर अन्य Modern Programming Languages के साथ सम्भव नहीं है। यदि हम Python Supported Devices के Categories की एक List बनाएं, तो हमारी List कुछ निम्नानुसार हो सकती है-

- Linux and Unix systems
- Microsoft Windows (all modern flavors)
- Mac OS (both OS X and Classic)
- BeOS, OS/2, VMS, and QNX
- Cray Supercomputers and IBM Mainframes
- Real-Time Systems
- PDAs running Palm OS, PocketPC
- Tablets and Smartphones running Google's Android and Apple's iOS
- Gaming Consoles and iPods
- Cell Phones running Symbian OS and Windows Mobile, etc...

Python Language Interpreter की तरह ही Python के साथ Ship होने वाले जो **Standard Library Modules** हैं, उन्हें भी इस तरह से Implement किया जाता है ताकि वे Cross-Platform व Cross-Machine Architecture पर यथासम्भव Portable रहें। इसीलिए Python के Programs, ऐसे Bytecodes में Translate होते हैं, जो कि Platform व Architecture Independent होते हैं और किसी भी Platform या Architecture पर Installed Compatible Python Version पर आसानी से Run हो जाते हैं।

यानी जिन Python Programs को Core Python Language व Standard Python Libraries का प्रयोग करते हुए Develop किया गया है, वे Linux, Windows, Unix, MacOS जैसे किसी भी Operating

System व Intel, AMD Athlon जैसे किसी भी CPU Architecture पर बिना किसी परेशानी के समान रूप से Run हो जाते हैं।

Python is Powerful

Python एक **Hybrid Language** है जिसकी वजह से इसके Toolsets इसे **Traditional Scripting Languages** (*Tcl, Scheme, Perl*) व **System Development Languages** (*C, C++, Java*) के कहीं बीच में Place कर देते हैं।

परिणामस्वरूप Python, किसी Scripting Language द्वारा Provide की जाने वाली Simplicity व Ease of Use वाले सारे Features तो Provide करता ही है लेकिन साथ ही वे Features व Tools भी Provide करता है, जो कि Advanced Software Development के लिए जरूरी Compiled Languages में ही उपलब्ध होते हैं। जिसकी वजह से Python को Large-Scale Development Projects के लिए भी उसी तरह से Use किया जा सकता है, जिस तरह से C/C++, Java जैसी Languages को Use किया जाता है।

चलिए, हम Python द्वारा Provide किए जाने वाले कुछ Developer Specific Extra-Ordinary Features को थोड़ा विस्तार से समझते हैं।

Dynamically Typed

Python, C/C++, Java, C# की तरह एक **Strictly Typed Programming Language** नहीं है, बल्कि PHP, JavaScript की तरह एक **Dynamically Typed Scripting Language** है। इसीलिए Python Program Create करते समय हम किस तरह की Value के साथ काम कर रहे हैं और उस Value को किस तरह के Objects द्वारा Handle किया जाना है, इस बात का ध्यान Program Run करते समय Python Interpreter द्वारा स्वयं ही रख लिया जाता है।

अन्य शब्दों में कहें तो Python में **Variable** या **Type** Declare करने जैसा कोई Concept ही नहीं है क्योंकि Python में Data Types जैसा कुछ होता ही नहीं है। जो भी कुछ होता है, वो Object होता है और हर मान किसी न किसी तरह के Object द्वारा ही Represent होता है क्योंकि Python को मूलतः Object Oriented Concepts को ध्यान में रखते हुए ही Develop किया गया है और Object Oriented Programming में सबकुछ एक Object होता है।

Automatic Dynamic Memory Management

हम Python में जब भी किसी मान को Use करने के लिए Specify करते हैं, Python Interpreter स्वयं ही उस मान से सम्बंधित Object के लिए Appropriate Memory Allocate कर देता है, और जब भी उस Object की जरूरत नहीं रह जाती, Python Interpreter उस Object द्वारा Reserved Memory Space को Release कर देता है, जिसे **Automatic Garbage Collection** के नाम से जाना जाता है।

इतना ही नहीं, किसी Object के लिए जितना Space Allocate किया गया है, यदि उस Object द्वारा और ज्यादा Data को Store करने की जरूरत पड़े, तो Python Interpreter स्वयं ही उस Object के लिए जरूरत के अनुसार Memory को बढ़ाकर Object को Expand भी देता है और यदि Object में से कुछ Data को Remove कर दिए जाने की वजह से कुछ Space Free हो गया हो, तो उस Object के लिए Reserved उस अनावश्यक Space को Free करके Object को Shrink भी कर दिया जाता है।

Automatic Memory Allocation व **Garbage Collection** की ये सुविधा Modern Programming Language जैसे कि Java, C# आदि में उपलब्ध एक बहुत ही विशिष्ट प्रकार की सुविधा है जो हमें C/C++ जैसी Lower Level Languages में उपलब्ध नहीं होती हैं। परिणामस्वरूप सामान्यतः C/C++ में Memory Management का काम हमें हमारी जरूरत के अनुसार Manually करना पड़ता है और निश्चित रूप से बहुत Complex व Buggy काम होता है।

लेकिन क्योंकि Python, C/C++ के समकक्ष ही Lower Level Memory Details को Track व Manage कर सकता है, इसलिए ये हमारे लिए Memory Management का काम भी स्वयं ही Perform कर देता है, जिसके लिए हमें अलग से कुछ नहीं करना पड़ता।

Built-In Support for Large-Scale Application Development

जब हम Python का प्रयोग करते हुए Large Applications Develop करना चाहते हैं, तब हमें कुछ Special Tools की जरूरत पड़ती है, ताकि हम हमारे Large Application को Multi-Developer Environment में Develop कर सकें। Python एक Large Application Develop करने से सम्बंधित सभी जरूरी Tools जैसे कि Modules, Classes, Exceptions आदि Provide करता है जिनके माध्यम से हम हमारे Large Application को-

- ✓ छोटे-छोटे Components के रूप में Organize कर पाते हैं।
- ✓ OOPS का प्रयोग करते हुए Reusable Codes Develop कर पाते हैं।
- ✓ जरूरत के अनुसार उन OOPS आधारित Classes को Inherit करके Customize भी कर पाते हैं।
- ✓ Application से सम्बंधित विभिन्न प्रकार के Events को Handle कर पाते हैं और
- ✓ अपने Application से सम्बंधित विभिन्न प्रकार की Errors, Exception व Bugs को आसानी से Handle व Manage कर पाते हैं।

OOPS के के साथ ही Python हमें Procedural व Functional Programming करने की भी सुविधा प्रदान करता है, जिनकी वजह से हम हमारे Large Application में जरूरत पड़ने पर OOPS के साथ ही Python द्वारा Provide किए जाने वाले Procedural व Functional Programming Tools का भी Mixed तरीके से भी प्रयोग कर सकते हैं और किसी Specific प्रकार की समस्या का समाधान Develop कर सकते हैं।

Built-In Data-Structure Object Types

Python कई Common रूप से Use किए जाने वाले **List**, **Dictionary** व **String** जैसे Data-Structures Built-In रूप से Support करता है, जो कि बहुत ही Flexible व Easy to Use होते हैं और हम अपनी ज्यादातर जरूरतों को पूरा करने के लिए आसानी से इनका प्रयोग कर सकते हैं।

ये Built-In रूप से Supported Data Structures इतने Well Optimized हैं कि उतना ही Memory Use करते हैं, जितने कि Currently पर जरूरत होती है और उतने ही समय के लिए Reserve करते हैं जितने की Current Time पर जरूरत होती है। यानी ये Data Structures, जरूरत के अनुसार Automatically Grow व Shrink होते रहते हैं, जिसके लिए हमें अलग से कुछ भी नहीं करना पड़ता।

उदाहरण के लिए यदि किसी List में 10 Data Stored हैं और हम किसी एक Data को Delete कर दें, तो तुरन्त ही Python केवल 9 Data के लिए ही Memory Space Reserved रखेगा और 1 Delete कर दिए गए Data के लिए Reserved Memory Space को तुरन्त Release कर देगा। जबकि यदि हम उसी List में एक और नया Data Add कर दें, तो List Automatically केवल एक और नए Data के लिए Memory Reserve कर लेगा। जबकि Memory Management से सम्बंधित ये सारी प्रक्रिया Dynamically अपने आप Internally Perform होती रहेगी, जिसके लिए हमें अलग से कुछ भी नहीं करना होगा।

Python द्वारा Provided सभी Built-In Data Structures इतने Flexible होते हैं, कि हम इन्हें अपनी जरूरत के अनुसार जैसे चाहें वैसे Nest कर सकते हैं यानी एक Data Structure के अन्दर दूसरे Data Structure को Embed कर सकते हैं और किसी Complex Data को भी आसानी से Represent कर सकते हैं।

Data-Structure Object Types Built-In Access and Manipulation Support

Python हमें जितने भी Built-In **Object Types** (*List, Dictionary, String, etc...*) Provide करता है, उन पर जितने भी तरह के Standard Operations Perform किए जा सकते हैं, उन सभी Operations को Perform करने से सम्बंधित सभी जरूरी तरीके भी Provide करता है।

उदाहरण के लिए हम इन Objects को **Concatenate** कर सकते हैं, इनकी **Slicing** कर सकते हैं, इनकी **Sorting** कर सकते हैं, इनकी **Mapping** कर सकते हैं और इन सभी Standard Tasks को Accomplish करने के लिए हमें अलग से कोई Code Create नहीं करना होता, बल्कि Python इन सभी कामों को Perform करने से सम्बंधित Functionalities हमें Built-In Functions के रूप में Provide कर देता है।

Standard Core-Libraries Support

जब हमें और भी Specific तरह के Task Perform करने होते हैं, तो उन Tasks को Perform करने के लिए Python हमें Pre-coded Standard Library Tools का एक Large Collection भी Provide करता है, जो कि Regular Expression Matching से लेकर Networking तक की सभी जरूरतों को आसानी पूरा करने में सहायक होता है।

जब एक बार हम Python Language को ठीक से समझ लेते हैं, उसके बाद हमें अपने Application Level की विभिन्न प्रकार की जरूरतों को पूरा करने के लिए केवल Python की **Core Libraries** व Third-Party द्वारा Provide की जाने वाली **External Libraries** को ही अपनी जरूरत के अनुसार समझना व Use करना होता है।

Third-Party External Libraries Support

चूंकि Python एक Open Source Language है, इसलिए Python Developers केवल Python की Core-Library तक ही सीमित नहीं रह सकते बल्कि उन्हें Pre-coded Third-Party Libraries को भी अपनी जरूरत के अनुसार समझना व उपयोग में लेना सीखना होता है। क्योंकि Python Core Library में हालांकि बहुत कुछ है, लेकिन Core Library द्वारा हम सभी तरह की जरूरतें पूरी नहीं कर सकते।

उदाहरण के लिए Python, Web Applications Develop करने के लिए CGI Support के अलावा Directly कोई Library Support Provide नहीं करता और वर्तमान समय में हम CGI Support पर आधारित Web Applications Develop नहीं कर सकते। इसलिए इस तरह के Web Applications Develop करने के लिए हमें Flask, Django जैसी Third-Party Web Application Development Framework का प्रयोग करना जरूरी हो जाता है।

इसी तरह से **COM Support, Imaging, Numeric Programming, XML, Database Access** आदि विभिन्न प्रकार की जरूरतों को पूरा करने के लिए हमें विभिन्न प्रकार की Third-Party Python Library का प्रयोग करना जरूरी हो जाता है।

Python is Multi-Language Supported

Python Programs को आसानी से अन्य Programming Languages में Develop किए गए Components के साथ Mix करके Use कर सकते हैं।

उदाहरण के लिए Python का **C API** हमें ये सुविधा देता है कि हम किसी Python Program में C Program को Call कर सकते हैं। इसका मतलब ये हुआ कि हम हमारी Special प्रकार की Performance Intensive जरूरत को पूरा करने के लिए अपनी Python Script में जहां चाहें, वहां C Language में Developed Compiled Codes को Embed कर सकते हैं और जहां चाहें वहां, Python Program को

अन्य Environment या System में Embed करके उस System की Functionality को Extend कर सकते हैं।

C/C++ जैसी Language में Developed Code Library को Python के साथ Mix कर सकने की सुविधा के कारण हम Python को एक Easy to Use Frontend Language व Customization Tool के रूप में Use कर सकते हैं। जिसके परिणामस्वरूप ही हम Python को एक Rapid Prototyping Development Tool की तरह Use कर पाते हैं, जहां किसी भी जटिल System को पहले Python में Develop किया जाता है और फिर उस System के केवल Performance Intensive Tasks को Accomplish करने वाले Components को C/C++ जैसी Compiled Language में Develop करके Python में Embed कर लिया जाता है।

Python is Easy to Learn and Use

Python को सीखना व Use करना C/C++, Java, C# जैसी किसी भी अन्य Language को सीखने व Use करने की तुलना में कहीं ज्यादा आसान है। बस हमें Python Code को Type करना होता है और Run करना होता है। इसके अलावा अन्य Programming Languages की तरह हमें Python Program को Compile and Link करने से सम्बंधित Lengthy Task को Follow करना जरूरी नहीं होता।

Python Interpreter, Programs को तुरन्त Run कर देता है, जिसकी वजह से एक Python Programmer अपना पूरा ध्यान अपने Solution को Develop करने पर लगा सकता है। साथ ही Python Programming के Syntax भी काफी आसान होते हैं और विभिन्न प्रकार की जरूरतों को आसानी से पूरा करने के लिए Python हमें ढेर सारे Built-In Tools भी Provide करता है, जिनकी वजह से Python का प्रयोग करते हुए Application Develop करना काफी आसान हो जाता है।

यदि आपने C/C++, Java, C# में से कोई Programming Language सीखी है, तो Python को सीखना आपके लिए बिल्कुल आसान हो जाएगा क्योंकि ये सभी Programming Languages, C Language के Syntax को ही Inherit करते हैं।

Python v/s Others

इस Chapter में हमने Python की कई विशेषताओं का वर्णन किया है, लेकिन इसका मतलब ये नहीं है कि Python ही दुनियाँ की सबसे अच्छी Language है। एक Programmer के रूप में हम हमारे Career में समय-समय पर कई Languages सीखते हैं और Different प्रकार के Projects के लिए अलग-अलग Languages को Use करते हैं।

इसका मतलब ये बिल्कुल भी नहीं है कि कोई एक Programming Language किसी दूसरी से ज्यादा बेहतर है। हर Programming Language की अपनी कुछ विशेषताएँ होती हैं, तो कुछ कमियाँ भी होती हैं

और हमें हमारे Current Project के आधार पर इस बात का निर्णय लेना होता है कि Current Project के लिए कौनसी Language सबसे अच्छी रहेगी।

इसलिए एक Programmer के रूप में आपको Multiple Languages सीखनी चाहिए ताकि आप इस बात का निर्णय ले सकें कि किस तरह की जरूरत को पूरा करने के लिए कौनसी Language सबसे बेहतर रहती है।

PYTHON

Interpreter

Internal Working

Python – Interpreter Internal Working

इस Chapter में हम Python Program के Execution के बारे में विस्तार से जानेंगे और समझने की कोशिश करेंगे कि एक Python Program किस तरह से Execute व Run होता है और किस तरह से Output Generate करता है।

इसके साथ ही हम Python Program Develop करने से सम्बंधित IDE, Text Editors के बारे में भी संक्षेप में Discuss करेंगे, ताकि आप अपने Development के लिए लिए उपयुक्त IDE या Text Editor का चुनाव कर सकें और आसानी से अपने Python Program को Develop व Run कर सकें।

Python Interpreter

पिछले Chapter में भी हमने Python को एक Programming Language के रूप Discuss किया था। Python मूलतः एक Interpreter है जो कि वास्तव में एक ऐसा Software Package होता है जो किसी अन्य Program को Execute करता है। जब हम Python Program लिखते हैं, तब Python Interpreter उस Program की प्रत्येक Line को One by One Read करता है और उनसे सम्बंधित Operations को Accomplish करता है। इसलिए Python Interpreter को हम अपने Python Program व Computer Hardware Machine के बीच का Software Logic Layer कह सकते हैं।

जब Python Package हमारे Computer System पर Install किया जाता है, तब ये हमारे Computer System पर कम से कम एक Interpreter और एक सम्पूर्ण Package Library के साथ Install होता है। Python कई तरह की जरूरतों के आधार पर कई तरह के Flavors में उपलब्ध है इसलिए हम इसे किस रूप में Install करते हैं उसके आधार पर इसकी Different Libraries का Set Install होता है।

यानी हम Python को Executable Program के रूप में भी Install कर सकते हैं और किसी अन्य Program की Supporting Linked Library के रूप में भी Install कर सकते हैं, जो कि पूरी तरह से हमारी जरूरत पर निर्भर करता है। हम इसे C Program के रूप में भी Install कर सकते हैं, Java Class के रूप में भी Install कर सकते हैं अथवा किसी अन्य तरह के Implementation के रूप में भी Install कर सकते हैं लेकिन Default रूप से हम जिस Python Version को Install करते हैं, वह मूलतः C के Implementation पर आधारित CPython ही होता है।

हम चाहे जिस भी तरह का Python Implementation Use करें, ये हमेशा हमारे Python Codes को Interpret करने का ही काम करता है और ये हमारे Python Program को Interpret कर सके, इसके लिए जरूरी है कि हम Python Interpreter को अपने Computer System पर Install करें।

विभिन्न प्रकार के Operating Systems के लिए Python के अलग-अलग तरह के Version उपलब्ध हैं जिन्हें हम Internet से Download करके अपने Computer System पर Install कर सकते हैं जबकि

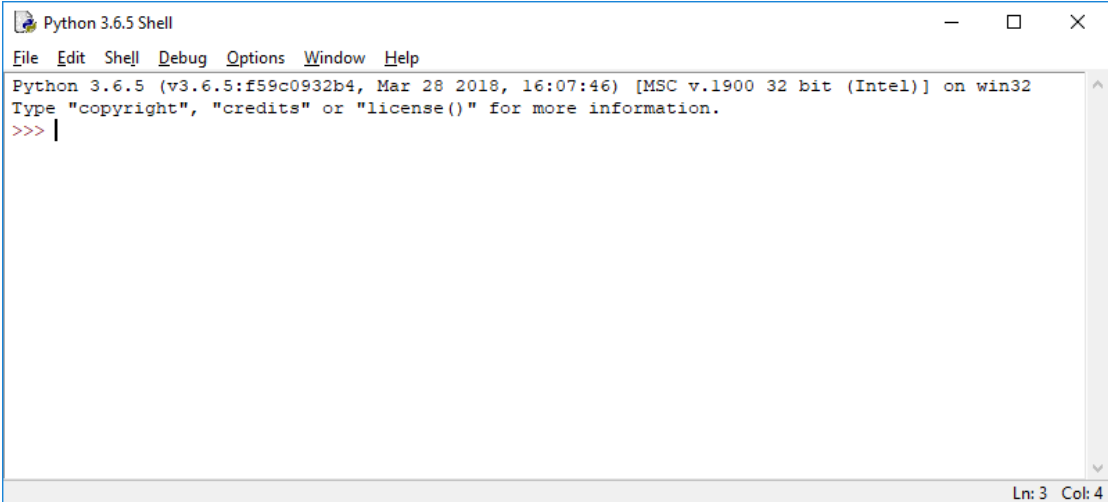
Python को Different Operating System पर Install करने का पूरा Process Internet पर आसानी से प्राप्त करके Follow किया जा सकता है।

इसलिए हम यहां पर Python Install करने की प्रक्रिया को Discuss नहीं करेंगे बल्कि हम यही मान कर चलेंगे कि आपने अपने Windows आधारित Computer System पर Python को **C:\python** Path पर Install कर लिया है और ये Path आपके Computer System के "**Path**" Environment Variable में Add हो गया है, जो कि Python Installation के दौरान अपने आप ही Set हो जाता है। इसलिए आप अपने Computer System पर कहीं से भी Python Interpreter को Invoke कर सकते हैं।

हालांकि <https://www.python.org>, Python Interpreter की मुख्य Website है, जहां से Python को Download करके Install किया जा सकता है लेकिन इसके अलावा भी Python के कई अलग Distributions भी उपलब्ध हैं, जिन्हें आप अपनी जरूरत के अनुसार Download करके Install कर सकते हैं।

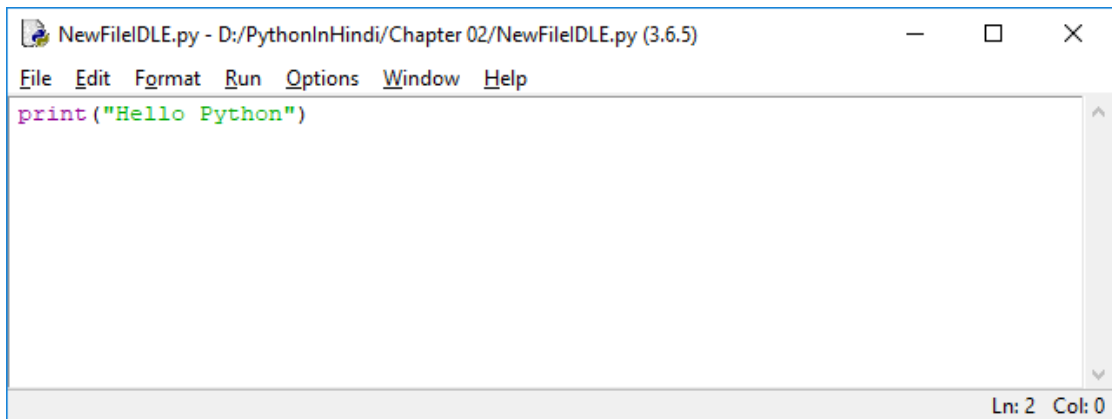
IDLE = Integrated Development and Learning Environment

जब हम Python Interpreter Install करते हैं, तब Python Package के साथ ही एक Default Editor भी Install होता है जो कि Python Shell होता है। इस Python Shell के माध्यम से हम सम्पूर्ण Python Program को भी Create करके Execute कर सकते हैं और किसी एक Single Python Statement को भी Execute करके उसका भी Output देख सकते हैं। ये IDLE Python Shell निम्नानुसार दिखाई देता है-



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
Ln: 3 Col: 4
```

इस IDLE के File Menu से New Option को Select करके हम एक New File के रूप में Python Script File Create कर सकते हैं, जिसका Extension .py होता है और फिर उस Python Script File को इसी Python Shell के माध्यम से Execute भी कर सकते हैं। जैसे-

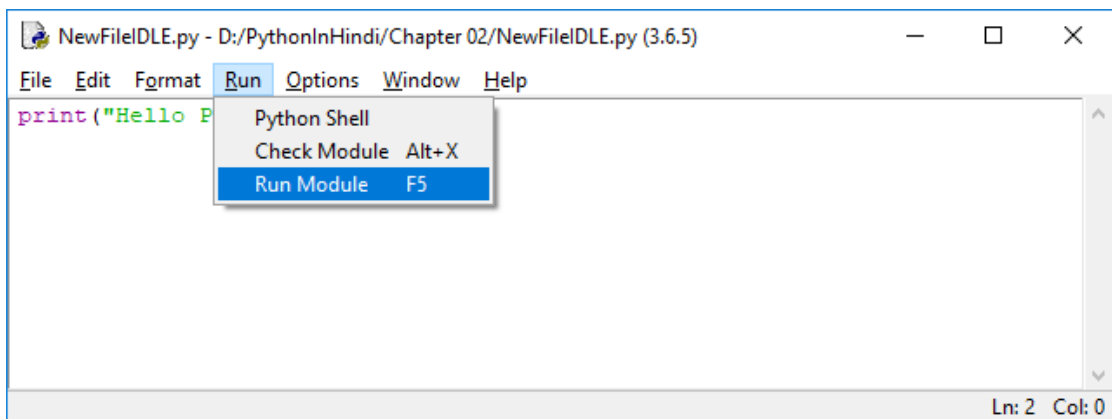


```

NewFileIDLE.py - D:/PythonInHindi/Chapter 02/NewFileIDLE.py (3.6.5)
File Edit Format Run Options Window Help
print("Hello Python")
Ln: 2 Col: 0

```

और फिर निम्न चित्रानुसार इसी File के **Run** Menu में दिखाई देने वाले **"Run Module F5"** Menu Option को Click करके अथवा Keyboard से **F5** Function Key Press करके उस Script File को Execute भी कर सकते हैं-

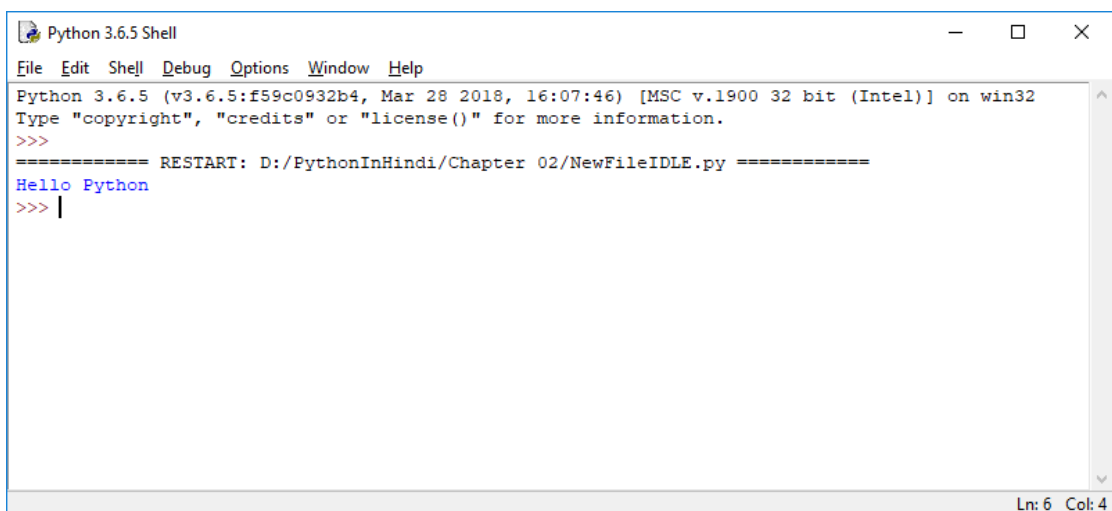


```

NewFileIDLE.py - D:/PythonInHindi/Chapter 02/NewFileIDLE.py (3.6.5)
File Edit Format Run Options Window Help
print("Hello P
Python Shell
Check Module Alt+X
Run Module F5
Ln: 2 Col: 0

```

परिणामस्वरूप जैसे ही ये File Run होती है, हमें Python Shell में निम्न चित्रानुसार Output भी दिखाई देने लगता है-

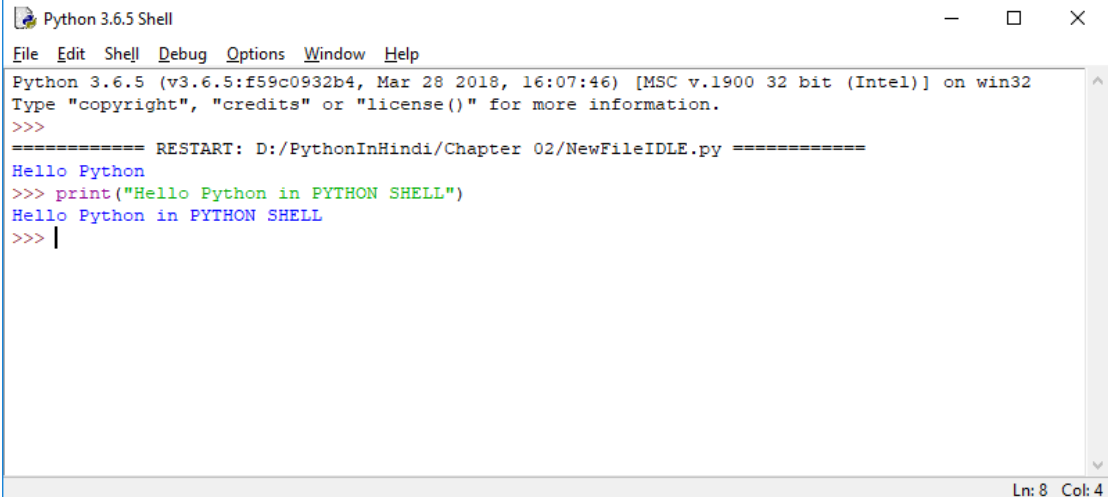


```

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/PythonInHindi/Chapter 02/NewFileIDLE.py =====
Hello Python
>>> |
Ln: 6 Col: 4

```

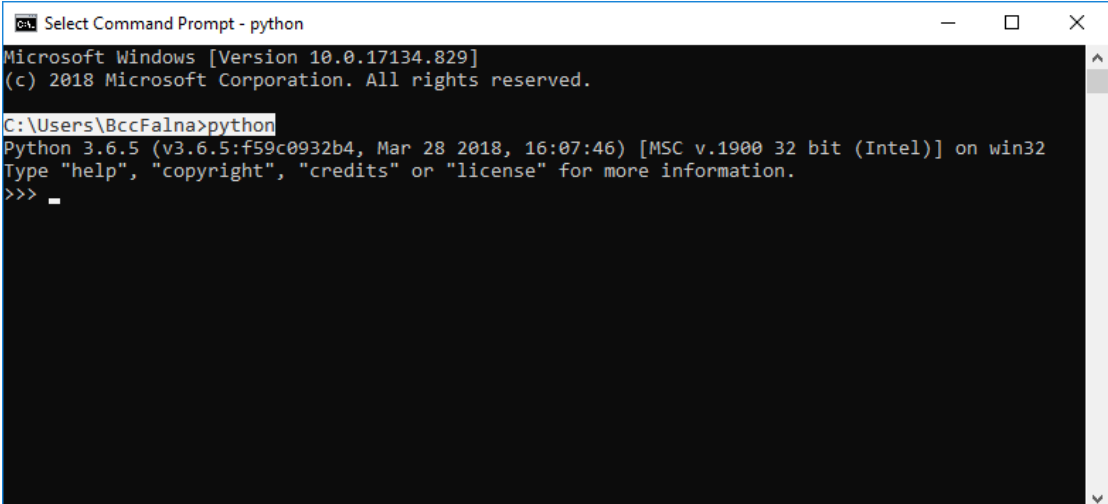
जबकि यदि हम चाहें तो किसी भी Python Statement को इसी Python Shell में लिखकर Directly भी Execute कर सकते हैं। जैसे-



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/PythonInHindi/Chapter 02/NewFileIDLE.py =====
Hello Python
>>> print("Hello Python in PYTHON SHELL")
Hello Python in PYTHON SHELL
>>> |
Ln: 8 Col: 4
```

Python Shell in Command Line

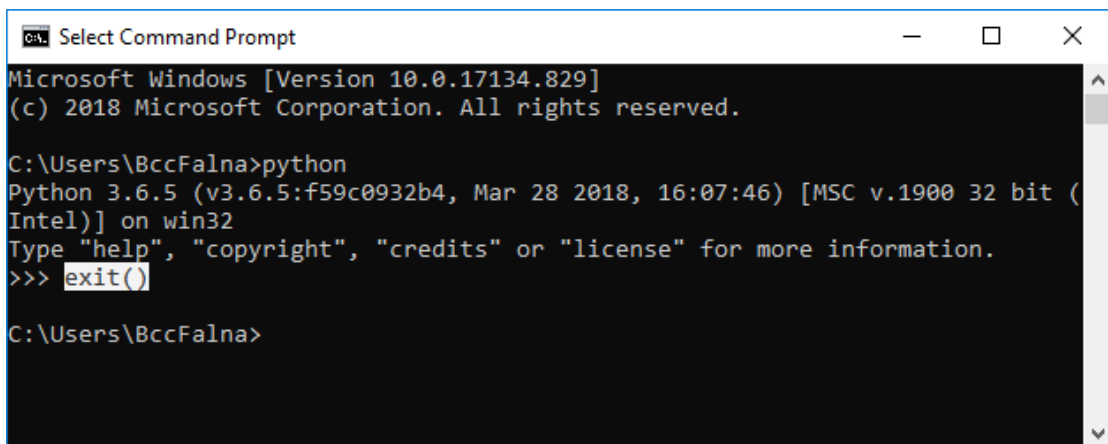
हम जब चाहें तब Command Prompt द्वारा **Python Shell** को Activate कर सकते हैं, जिसके लिए हमें Command Prompt में निम्न चित्रानुसार केवल **python** Type करके Enter करना होता है-



```
Select Command Prompt - python
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\BccFalna>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> _
```

एक बार **Python Shell** में Entry करने के बाद हम हर उस Python Command को यहां से भी Execute कर सकते हैं, जिसे IDLE द्वारा कर सकते हैं। जबकि इस Python Shell से Exit करने के लिए हमें **exit()** Statement लिखकर **Enter Key** Press करना होता है। जैसे:

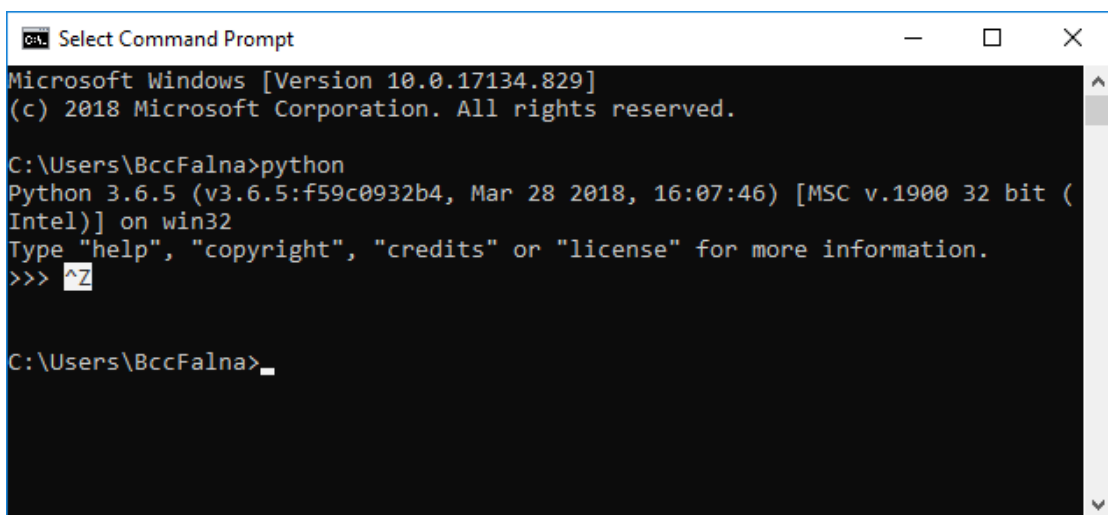


```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\BccFalna>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\BccFalna>
```

या फिर हमें **Ctrl + Z** Key Combination Press करके Entry Key Press करना होता है। जैसे-



```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

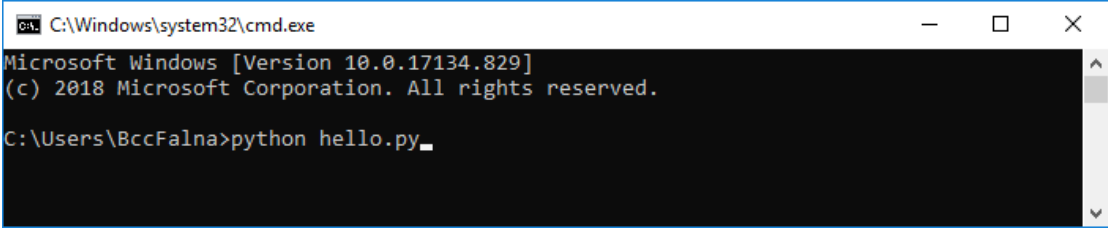
C:\Users\BccFalna>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ^Z

C:\Users\BccFalna>
```

Text Editors

IDLE व **Python Shell** के अलावा हम किसी भी अन्य **Text Editors** जैसे कि **Notepad**, **Notepad++**, **VS Code**, **Adom**, **Brackets**, **Sublime**, **VS Codium** आदि किसी भी Text Editor का प्रयोग करते हुए Python Script लिख सकते हैं। बस उस Script File को Save करते समय हमें उसका **Extension .py** रखना होता है।

एक बार Python Script File में Python Code लिख लेने के बाद हमें Python Interpreter को Command Prompt पर उस File का नाम Parameter के रूप में बताना होता है, ताकि Python Interpreter उस Script File के Codes को Execute कर सके और ये काम हम निम्नानुसार करते हैं-



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\BccFalna>python hello.py_
```

जहाँ hello.py वह Python Script File है जिसे Python Interpreter द्वारा Execute किया जाना है।

Interactive Prompt and Python Shell

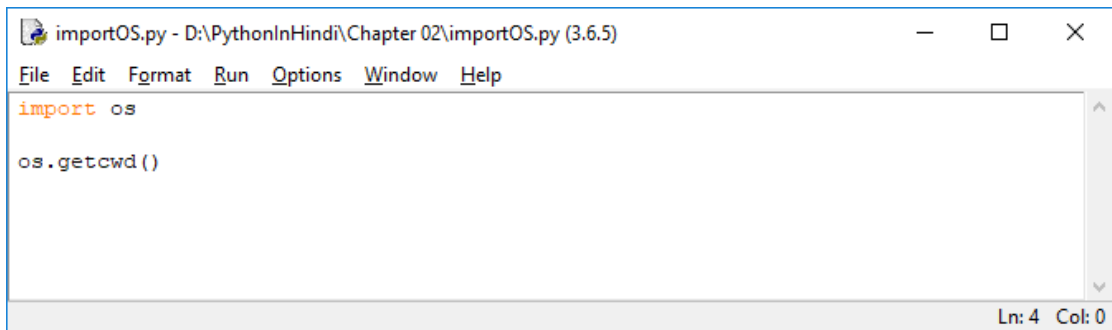
Python Interactive Prompt व **Python Shell** में केवल एक ही अन्तर है कि Python Script File में लिखे Python Codes को हम बिना दोबारा Type किए हुए जरूरत के अनुसार बार-बार Python Shell के माध्यम से Execute कर सकते हैं, लेकिन जब हम Interactive Prompt का प्रयोग करते हुए अपने Python Codes Execute करते हैं, तब हमें हमारे Python Codes को बार-बार Type करना पड़ता है क्योंकि Interactive Prompt हमारे Codes को किसी भी तरह से Save करके नहीं रखता।

इसीलिए Interactive Python Shell का प्रयोग केवल Codes के साथ Experiments करने और उन्हें Test करने के लिए ही किया जाता है। यानी जब भी कभी हमें हमारे Python Code के द्वारा Return होने वाले Output पर कोई शंका हो, तब उस Code को Interactive Prompt पर Execute करके हम अपने Python Code के Output को Test कर सकते हैं। लेकिन जब Codes को Finally Use करना होता है, तब उन्हें किसी Python Script File में ही लिखा जाता है और Python Shell पर उस File को ही Run किया जाता है।

हम जो भी Python Code किसी Python Script File में लिख सकते हैं, हर उस Code Statement को हम उसी Sequence में Interactive Prompt पर भी लिख सकते हैं और Interactive Prompt भी हमें Exactly वही Output देता है, जो उस Python Script File को Run करने पर मिलता है।

इसलिए यदि हमने हमारे Python Script File में किसी Module को Import करके उसके किसी Function को Call किया है, तो हम ठीक उसी तरह से Interactive Prompt में भी उसी Module को उसी तरह से Import कर सकते हैं और बिलकुल समान प्रकार से उस Module के किसी Function को भी Interactive Prompt पर Call कर सकते हैं।

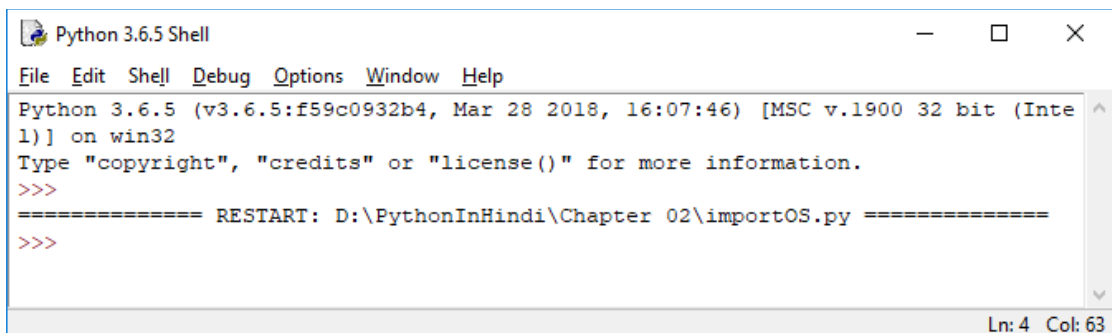
उदाहरण के लिए यदि हमें ये जानना हो कि हमारा Current Python Script File हमारे Computer System के किस Path पर Exist है, तो ये जानने के लिए हमें Python द्वारा Provide किए जाने वाले **os** Module के **getcwd()** Method को Invoke करना पड़ता है। इस स्थिति में हम किसी **importOS.py** नाम की Python Script File में निम्नानुसार Python Code लिख सकते हैं-



```
import os

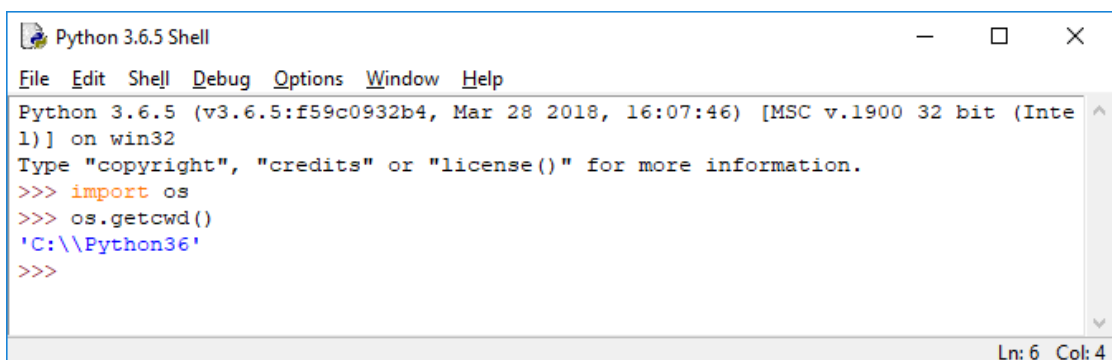
os.getcwd()
```

और जब हम इस Python Script File को Python Shell पर Run करने के लिए F5 Function Key Press करते हैं, तो हमें निम्नानुसार Current Python File की Location का पूरा Path (*D:\PythonInHindi\Chapter 02\importOS.py*) Output के रूप में प्राप्त होता है-



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\PythonInHindi\Chapter 02\importOS.py =====
>>>
```

लेकिन Exactly इसी Code को जब हम निम्नानुसार Interactive Shell पर Type करते हैं, तब भी हमें Current File जो कि Interactive Shell File है, का Path (*C:\\Python36*) Output के रूप में Return होता है-



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import os
>>> os.getcwd()
'C:\\Python36'
>>>
```

यहां समझने वाली बात ये है कि Interactive Prompt वह स्थान है, जो हमें हमारे Python Codes को Test करने की सुविधा देता है। इसलिए एक प्रकार से हम इसे हमारे Debugging Program की तरह भी Use करते हैं। क्योंकि जब कोई Python Code हमारा वांछित Output नहीं दे रहा होता है, तब उस Code को इसी Interactive Prompt के माध्यम से Test करके हम उसकी गलतियों का पता लगा सकते हैं।

जिस तरह से हमने उपरोक्त उदाहरण में **Interactive Prompt** के माध्यम से **os** Module को Import करके उसके `getcwd()` Method को Use किया है, ठीक उसी तरह से हम किसी Class या Function को भी Import कर सकते हैं और उसकी Functionality को Test कर सकते हैं। यानी-

- ✓ हम CPython के Interactive Prompt पर C/C++ के Class, Function, Modules को Import कर सकते हैं।
- ✓ हम Jython के Interactive Prompt पर Java के Class, Function, Modules को Import कर सकते हैं और इसी तरह से
- ✓ हम ActivePython के Interactive Prompt पर .NET Framework की Class, Function, Modules को Import कर सकते हैं।

Python के इसी Interactive Programming Style की वजह से Python Development सीखना और करना दोनों काफी आसान व सुविधापूर्ण हो जाता है।

हालांकि Python Interactive Prompt काफी उपयोगी व सुविधापूर्ण है लेकिन फिर भी इसके साथ अग्रानुसार कुछ Restrictions व Limitations हैं।

Only Python Commands in Interactive Prompt

हम इसमें सभी तरह के Python Statements व Commands का प्रयोग नहीं कर सकते जिनमें से एक सबसे महत्वपूर्ण **System Commands** हैं।

यानी हम Interactive Prompt पर System Commands, वे Commands जिन्हें DOS Prompt पर अथवा Linux/Unix के Shell Prompt पर Execute करते हैं, उन System Commands को Python के Interactive Prompt पर Execute नहीं कर सकते हैं। बल्कि इन System Commands को Execute करने के लिए हमें Python के **os** Module को Import करना पड़ता है और इस os Module द्वारा Provide किए जाने वाले Methods के माध्यम से ही हम किसी System Command को Interactive Prompt पर Execute कर सकते हैं, जैसाकि पिछले Example में किया है।

No print() Statement in Interactive Prompt

इतना ही नहीं, `print()` Statement का प्रयोग Results को Output में Display करने के लिए केवल Python Script File में ही किया जाता है। Interactive Prompt पर हमें `print()` Statement को Use करके Output को Display करना जरूरी नहीं होता क्योंकि Interactive Prompt हर Python Statement के Execution पर Return होने वाले Output को स्वयं ही Display कर देता है।

No Indentation in Interactive Prompt or Python Script File

Python में Indentation का अपना Specific महत्व है, इसलिए हम कभी भी Python Codes लिखते समय Python Script File में या Interactive Prompt में, कहीं पर भी Program Codes को सुन्दर व व्यवस्थित दिखाने के लिए Indentation नहीं कर सकते क्योंकि Python में Code Blocks (*Group of Codes*) की Nesting को Represent करने का काम Indenting द्वारा किया जाता है।

Code Blocks के Separation का यही काम C/C++, Java, C#, JavaScript जैसी अन्य Languages में Curly Braces द्वारा किया जाता है, लेकिन Python में Curly Braces द्वारा **Dictionary** नाम के एक Data Structure को Represent किया जाता है। इसीलिए Code Blocks की Nesting को Represent करने के लिए Python में Indenting का प्रयोग करते हैं।

यदि हम गलती से अपने किसी भी Code की Indenting कर दें, तो Python Shell व Interactive Interpreter हमें **“SyntaxError” Message** Return करते हैं। यहां समझने वाली बात ये भी है कि किसी भी Statement से पहले Indenting करने के लिए Tabs का प्रयोग करना ही जरूरी नहीं है। किसी भी Python Statement से पहले एक Single Space भी Indenting ही माना जाता है। इसलिए Code लिखते समय हमेशा ध्यान रखें कि जब तक Code Blocks की Nesting न हो रही हो, किसी भी Statement से पहले एक भी Blank Space या Tab नहीं होना चाहिए।

Only One Statement Runs at a Time in Interactive Prompt

Python Interactive Prompt पर एक समय में केवल एक ही Line Run होता है। इसलिए जैसे ही हम किसी Statement को Type करके Enter Key Press करते हैं, हमारा Last Python Statement Execute होकर Output Display कर देता है। लेकिन जब हम Multiple Lines के Codes को Execute करना चाहते हैं, तब Best तरीका यही होता है कि हम उन्हें एक Python Script File में लिखकर Python Shell के माध्यम से Execute करें।

Multiline Compound Statements in Interactive Prompt

हालांकि जब हम **Conditional, Looping** या **Branching** Statements को Interactive Prompt पर Test करना चाहते हैं, तब हमें Multiline Compound Statement लिखना पड़ता है और Indenting के माध्यम से Codes Blocks को Nest भी करना पड़ता है। इस जरूरत को सामान्यतः निम्नानुसार तरीके से पूरा किया जाता है-

How to Get Complete PDF EBook

आप **Online Order** करके **Online** या **Offline** Payment करते हुए इस Complete EBook को तुरन्त Download कर सकते हैं।

Order करने और पुस्तक को Online/Offline Payment करते हुए खरीदने की पूरी प्रक्रिया की विस्तृत जानकारी प्राप्त करने के लिए आप BccFalna.com के निम्न Menu Options को Check Visit कर सकते हैं।

How to Make Order

[How to Order?](#)

How to Buy Online

[How to Pay Online using PayUMoney](#)

[How to Pay Online using Instamojo](#)

[How to Pay Online using CCAvenue](#)

How to Buy Offline

[How to Pay Offline](#)

[Bank A/c Details](#)

जबकि हमारे Old Buyers के [Reviews](#) भी देख सकते हैं ताकि आप इस बात का निर्णय ले सकें कि हमारे Buyers हमारे PDF EBooks से कितने Satisfied हैं और यदि आप एक से अधिक EBooks खरीदते हैं, तो [Extra Discount](#) की Details भी Menubar से प्राप्त कर सकते हैं।