

# Python for Everyone

---

*Learn and polish your coding  
skills in Python*

---

**Saurabh Chandrakar  
Dr. Nilesh Bhaskarrao Bahadure**



[www.bpbonline.com](http://www.bpbonline.com)

Copyright © 2023 BPB Online

*All rights reserved.* No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor BPB Online or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

BPB Online has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, BPB Online cannot guarantee the accuracy of this information.

First published: 2023

Published by BPB Online

WeWork

119 Marylebone Road

London NW1 5PU

**UK | UAE | INDIA | SINGAPORE**

ISBN 978-93-55518-170

[www.bpbonline.com](http://www.bpbonline.com)

## **Dedicated to**

### *Dedicated to my Parents*

*Dr Surendra Kumar Chandrakar and  
Smt. Bhuneshwari Chandrakar*

*brother*

*Shri Pranav Chandrakar*

*to my wife*

*Priyanka Chandrakar*

*and to my lovely son*

*Yathartha Chandrakar*

*Saurabh Chandrakar*

### *Dedicated to my Parents*

*Smt. Kamal B. Bahadure and Late Bhaskarrao M. Bahadure  
to my in-laws*

*Smt. Saroj R. Lokhande and Shri. Ravikant A. Lokhande  
and to*

*my wife Shilpa N. Bahadure and to beautiful daughters*

*Nishita and Mrunmayee*

*And to all my beloved students.*

*Dr. Nilesh Bhaskarrao Bahadure*

---

## About the Authors

- **Saurabh Chandrakar** is a Research & Development Engineer (Dy. Manager) at Bharat Heavy Electricals Limited (BHEL) Hyderabad. He is the winner of the best executive award on Operations Division by BHEL Hyderabad. Recently, he has been awarded the prestigious BHEL Excellence Award under Anusandhan category for Redundant Composite Monitoring System of Power Transformers project. He has 20 copyrights and 1 patent granted. Additionally, he has 6 patents filed. Moreover, he has published 3 books in reputed publications such as BPB New Delhi (Programming Techniques using Python), Scitech Publications Chennai (Programming Techniques using matlab) and IK International publishers (Microcontrollers and Embedded System Design). He has also launched 1 video course on BPB titled “First Time Play with Basic, Advanced Python concepts and complete guide for different python certification exams all in one umbrella.”
- **Dr. Nilesh Bhaskarrao Bahadure** received the Bachelor of Engineering degree in Electronics Engineering in 2000, the Master of Engineering degree in Digital Electronics in 2005, and the Ph.D. degree in Electronics in 2017 from KIIT Deemed to be University, Bhubaneswar, India. He is currently an Associate Professor in the Department of Computer Science and Engineering at Symbiosis Institute of Technology (SIT), Nagpur, Symbiosis International (Deemed University) (SIU), Pune, Maharashtra, India. He has more than 20 years of experience. Dr. Bahadure is a life member of IE(I), IETE, ISTE, ISCA, SESI, ISRS, and IAENG professional organizations. He has published more than 40 articles in reputed international journals and conferences, and has 5 books to his credit. He is the reviewer of many indexed journals such as IEEE Access, IET, Springer, Elsevier, Wiley and so on. His research interests are in the areas of Sensor Technology, the Internet of Things, and Biomedical Image Processing

---

## Acknowledgements

First and foremost, I would like to thank you all for selecting this book. It has been written with the beginner reader in mind. First of all, I take this opportunity to greet and thank my mentor "Prof. Nilesh Bahadure Sir" for motivating me and always communicating his expertise fully on topics related to Python. I am very thankful of being his protégé. I appreciate his belief in me, for always standing behind me, and pushing me to achieve more. The phrase "Journey of Thousand Miles Begins with a Single Step" is something he always reminds me of.

My parents, Dr. Surendra Kumar Chandrakar and Smt. Bhuneshwari Chandrakar, my brother, Shri Pranav Chandrakar, my beloved wife, Mrs. Priyanka Chandrakar, my adorable son, Yathartha Chandrakar, and all of my friends have inspired me and given me confidence over the years. Last but not least, I would like to express my sincere gratitude to the staff at "BPB Publications Private Limited" for their contributions and insights that made parts of this book possible.

— *Saurabh Chandrakar*

It was my privilege to thank Dr. S. B. Mujumdar, Chancellor of the Symbiosis International University, Pune and Shri. Vijay Kumar Gupta, Chairman of Beekay Industries Bhilai and BIT Trust, for his encouragement and support. I would like to thank my mentors Dr. Arun Kumar Ray, Dean, School of Electronics Engineering, KIIT Deemed to be University, Bhubaneswar and Dr. Anupam Shukla, Director, SVNIT Surat. I would like to thank Dr. Vidya Yeravdekar, Principal Director of Symbiosis Society, and the Pro Chancellor of Symbiosis International University, Pune, Dr. Rajani R. Gupte, Vice Chancellor of the Symbiosis International University, Pune, and Dr. Ketan Kotecha, Dean, Faculty of Engineering, Symbiosis International University, Pune, and Dr. Mukesh M. Raguwanshi, Director, SIT Nagpur, for their advice, and encouragement throughout the preparation of the book.

I would also like to thank Dr. Sanjeev Khandal, HOD, Department of Aeronautical Engineering, SGU Kolhapur, my well-wisher Dr. Prasenjeet D. Patil, Associate Professor, MIT ADT University, Pune and my colleagues in Symbiosis Institute of Technology Nagpur for providing valuable suggestions and lots of encouragement throughout the project.

I am thankful to Prof. Dr. N. Raju, Sr. Assistant Professor, SASTRA University, Thanjavur, Tamil Nadu, for his support, assistance during writing, and for his valuable suggestions.

I would also like to thank Dr. Ravi M. Potdar, Sr. Associate Professor, BIT Durg, and Dr. Md. Khwaja Mohiddin, Associate Professor, BIT Raipur for providing valuable suggestions and lots of encouragement throughout the project. Writing a beautiful, well balanced and acquainted book is not a work of one or two days or a month; it takes a lot of time and patience, as well as hours of hard work. Many thanks to my family members, my parents, wife, children and my well-wishers for their kind support. Without them and their faith and support, writing this classic book, would have remained just a dream. I also like to thank my students, who have always been with me, for relating problems and to finding solutions too. The perfection in any work does not come in a day. It needs a lot of effort, time and hard work, and sometimes, proper guidance.

It is my privilege to thank Prof. (Dr.) Ram Dhekekar, Professor, Department of Electronics & Telecommunication Engineering, SSGMCE Shegaon and Dr. C. G. Dethe, Director UGC Staff College Nagpur. Last but not the least, I would like to offer an extra special thanks to the staff at "BPB Publications Private Limited" for their insight and contribution in polishing this book.

Most significantly, I want to thank Lord Ganesha for all of the work I was able to put into the book's preparation. I would not be as zealous as I am now if it weren't for God's amazing creation of the universes.

*For since the creation of the world God's invisible qualities - his eternal power and divine nature - have been clearly seen, being understood from what has been made, so that men are without excuse."*

— **Dr. Nilesh Bhaskarrao Bahadure**

## Preface

The purpose of this book is to introduce readers with little to no programming experience to Python and provide them with the foundational knowledge and skills necessary to begin writing code in the language. By mastering Python, readers will be able to apply this technology to solve real-world problems and create useful applications.

The first part of the book covers core python concepts. Then we shall see some advanced Python concepts along with some data science libraries like numpy, matplotlib and pandas. Finally, in the later part of the book, we shall see some powerful observations while writing python code.

This book covers a wide range of topics, from basic syntax and data types to more advanced concepts along with a little touch of data science. Overall, the book provides a solid foundation for beginners to start their journey for getting trained in Python language.

This book is divided into 11 chapters. Each chapter's description is listed as follows.

**Chapter 1: Basic Python Introduction** – will cover the basic elements of writing a Python program where readers can gain an understanding of the fundamental topics.

**Chapter 2: Concept of Strings in Python** – will cover strings in detail, as well as different string methods, usage of command-line arguments, and string access with some examples.

**Chapter 3: Concept of Flow Control Statements in Python** – will cover the concept of conditional, iterative, and transfer statements, with loop patterns such as star pattern, alphabet pattern, and number pattern, with examples.

**Chapter 4: Concept of Exception Handling in Python** – will cover the concept of multiple ways of using the try-except block, for preventing errors during runtime. Users will be able to create their exceptions for handling the errors. Moreover, the user will see different use cases and control flow of Python exception hierarchy.

**Chapter 5: Concept of Regular Expressions in Python** – will cover the declarative mechanism for the representation of a group of strings, according to a particular pattern called regular expressions, which is quite difficult to understand, and also

perform its usage. Each regex expression is explained in a very lucid manner with examples.

**Chapter 6: Concept of Functions in Python** – will cover function types, and different types of function arguments such as positional arguments, keyword arguments, default arguments, variable length arguments, and kwargs. Moreover, concepts on local, global and non-local variable are well explained with examples, along with Python closures.

**Chapter 7: Concept of Data Structures in Python** – will cover non-primitive inbuilt data structures such as list, tuple, set and dictionary, which are unique on their own. All the data structures definitions, methods, and different types of comprehensions such as list comprehension, tuple comprehension, set comprehension, and dictionary comprehension are well discussed.

**Chapter 8: Concept of Packages in Python** – will cover the demonstration of package examples, with different approaches to module usage.

**Chapter 9: Numpy Introduction** – will cover different examples of scientific computing and data analysis library, that is, numpy library.

**Chapter 10: Data Visualization Introduction** – will cover Matplotlib data visualization library by creating a line plot with examples.

**Chapter 11: Pandas Introduction** – will cover Pandas Series and Pandas DataFrame with examples.



---

## Code Bundle and Coloured Images

Please follow the link to download the *Code Bundle* and the *Coloured Images* of the book:

**<https://rebrand.ly/33uiwxo>**

The code bundle for the book is also hosted on GitHub at **<https://github.com/bpbpublications/Python-for-Everyone>**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **<https://github.com/bpbpublications>**. Check them out!

## Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

**[errata@bpbonline.com](mailto:errata@bpbonline.com)**

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.bpbonline.com](http://www.bpbonline.com) and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

**[business@bpbonline.com](mailto:business@bpbonline.com)** for more details.

At **[www.bpbonline.com](http://www.bpbonline.com)**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

### Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at [business@bpbonline.com](mailto:business@bpbonline.com) with a link to the material.

### If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit [www.bpbonline.com](http://www.bpbonline.com). We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

### Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit [www.bpbonline.com](http://www.bpbonline.com).

## Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



---

# Table of Contents

<b>1. Basic Python Introduction.....</b>	<b>1</b>
Introduction .....	1
Structure .....	1
Objectives .....	2
Benefits of Python .....	2
Uses of Python .....	4
Limitations of Python .....	4
Keywords and reserved words .....	13
Identifiers.....	15
Line joining methods .....	15
<i>Implicit line joining method.....</i>	<i>16</i>
<i>Using curly braces .....</i>	<i>16</i>
<i>Using square brackets .....</i>	<i>16</i>
<i>Using parenthesis .....</i>	<i>16</i>
<i>Explicit joining method.....</i>	<i>17</i>
<i>Example 1 .....</i>	<i>17</i>
<i>Example 2 .....</i>	<i>18</i>
Print function.....	18
<i>Different styles to use print function.....</i>	<i>19</i>
Variables.....	24
Importance of mnemonic variable names in Python .....	27
Concept of immutability vs fundamental data types .....	29
Conclusion.....	33
Points to remember .....	33
Questions .....	33
<b>2. Concept of Strings in Python.....</b>	<b>35</b>
Introduction .....	35
Structure .....	36
Objectives .....	36
Reading dynamic input from the keyboard .....	36
<i>raw_input() .....</i>	<i>36</i>

---

<input/> () .....	37
Using Try Except .....	40
Using eval function.....	41
Command Line arguments .....	43
Python sys.argv.....	43
Python getopt module .....	46
Exception getopt.GetoptError .....	47
Short form options.....	47
Long form options.....	48
Python argparse module.....	49
Python argparse Positional arguments.....	50
Python argparse positional arguments default values .....	51
Python argparse argument help.....	52
Python argparse Data Type .....	54
Python argparse optional arguments.....	55
Short names for optional arguments with argparse.....	56
Python argparse with optional and positional arguments.....	57
Python argparse with required.....	57
Python argparse dest action.....	58
Python argparse append action .....	59
Allowing or disallowing abbreviations.....	59
Strings.....	61
Python multiline strings .....	62
Python string access .....	64
Using index .....	64
Using slice operator.....	65
In forward direction.....	66
In backward direction.....	66
Conclusion.....	71
Points to remember .....	71
Questions .....	71
<b>3. Concept of Flow Control Statements in Python .....</b>	<b>73</b>
Introduction .....	73
Structure .....	73

---

Objectives .....	74
Flow of execution of the program .....	74
<i>Selection statements/Conditional statements</i> .....	74
<i>if</i> .....	74
<i>if-else</i> .....	77
<i>if-elif-else</i> .....	79
<i>if-elif-else ladder</i> .....	81
<i>Iterative statements</i> .....	84
<i>for</i> .....	85
<i>while</i> .....	88
<i>Transfer statements</i> .....	91
<i>break</i> .....	91
<i>continue</i> .....	95
<i>pass</i> .....	100
in keyword usage .....	100
Loop patterns .....	102
<i>Star pattern</i> .....	102
<i>Printing stars in pyramid shape</i> .....	102
<i>Alphabet pattern</i> .....	104
<i>Number pattern</i> .....	107
Conclusion.....	109
Points to remember .....	109
Questions .....	110
<b>4. Concept of Exception Handling in Python .....</b>	<b>111</b>
Introduction .....	111
Structure .....	111
Objectives .....	112
Errors.....	112
<i>Syntax error</i> .....	112
<i>Runtime Error</i> .....	113
Importance of exception handling .....	114
Python exception hierarchy .....	115
Customized exception handling .....	116

---

Control flow in try-except .....	118
<i>Case-1 - No raising of exception</i> .....	119
<i>Case-2 - Exception raised at st3 and corresponding except block is matched</i> .....	119
<i>Case-3 - Exception raised at st3 and corresponding except block is not matched</i> .....	119
<i>Case-4 - Exception raised at st5</i> .....	119
<i>Case-5 - Exception raised at st6</i> .....	119
Exception information printing to the console .....	120
Try with multiple except blocks.....	121
Single except block that can handle multiple exceptions .....	122
Default except block.....	124
Possible combinations of except block.....	125
<i>finally except block</i> .....	126
<i>Control flow in try-except and finally</i> .....	129
<i>Case-1 - No exception is raised</i> .....	130
<i>Case-2 - Exception raised at st3 and corresponding except block is matched</i> .....	130
<i>Case-3 - Exception raised at st3 and corresponding except block is not matched</i> .....	130
<i>Case-4 - Exception raised at st5</i> .....	131
<i>Case-5 - Exception raised at st6 or st7</i> .....	131
<i>Nested try-except finally block</i> .....	131
<i>Control flow in Nested try-except finally block</i> .....	134
<i>Case-1 - If there is no exception</i> .....	135
<i>Case-2 - Exception raised at st3 and corresponding except block is matched</i> .....	135
<i>Case-3 - Exception raised at st3 and corresponding except block is not matched</i> .....	135
<i>Case-4 - Exception raised at st6 and inner except block is matched</i> ....	135
<i>Case-5 - Exception raised at st6 and inner except block is not matched but outer except block is matched</i> .....	135
<i>Case-6 - Exception raised at st6 and both inner and outer except block is not matched</i> .....	136
<i>Case-7 - Exception raised at st7 and the corresponding except block is matched</i> .....	136

---

Case-8 - Exception raised at st7 and the corresponding except block is not matched.....	136
Case-9 - Exception raised at st8 and the corresponding except block is matched .....	136
Case-10 - Exception raised at st8 and the corresponding except block is not matched.....	137
Case-11 - Exception raised at st9 and the corresponding except block is matched .....	137
Case-12 - Exception raised at st9 and the corresponding except block is not matched.....	137
Case-13 - Exception raised at st10.....	137
Case-14 - Exception raised at st11 or st12.....	138
Else block with try-except finally block.....	138
Conclusion.....	141
Points to remember .....	141
Questions .....	141
<b>5. Concept of Regular Expressions in Python.....</b>	<b>143</b>
Introduction .....	143
Structure .....	144
Objectives .....	144
compile().....	144
finditer().....	145
Character classes.....	147
Pre-defined character classes .....	153
Quantifiers.....	158
Functions of re module.....	162
<i>match</i> .....	162
<i>fullmatch</i> .....	164
<i>search</i> .....	166
<i>findall</i> .....	167
<i>sub</i> .....	168
<i>subn</i> .....	169
<i>split</i> .....	171
<i>escape</i> .....	173
Metacharacters.....	174

---

[] (Square brackets).....	174
. (Period).....	175
^ (Caret).....	175
\$ (Dollar).....	175
* (Star).....	176
+ (Plus).....	176
? (Question Mark).....	177
{} (Braces).....	177
(Alternation):.....	177
() (Group).....	178
\ (Backslash).....	178
r prefix.....	178
Conclusion.....	179
Points to remember.....	179
Questions.....	179
<b>6. Concept of Functions in Python.....</b>	<b>181</b>
Introduction.....	181
Structure.....	181
Objectives.....	182
Functions in Python.....	182
Function types.....	184
<i>Built in functions</i> .....	184
<i>User defined functions</i> .....	185
Function arguments.....	186
<i>Positional arguments</i> .....	187
<i>Keyword arguments</i> .....	189
<i>Default arguments</i> .....	190
<i>Variable length arguments</i> .....	192
<i>Keyword variable length arguments (kwargs)</i> .....	194
Nested function.....	197
Python closures.....	199
Function passing as a parameter.....	201
Local, global, and non-local variables.....	202
<i>Local variables</i> .....	202



---

<i>Global variables</i> .....	203
<i>Non-local variables</i> .....	207
Recursive function.....	208
Python Lambda functions .....	210
<i>Nested lambda functions</i> .....	211
<i>Passing lambda functions to another function</i> .....	212
Conclusion.....	213
Points of remember .....	213
Questions .....	214
<b>7. Concept of Data Structures in Python .....</b>	<b>215</b>
Introduction .....	215
Structure .....	215
Objectives .....	216
List data structure.....	216
<i>Creating a list</i> .....	216
<i>Creating an empty list</i> .....	216
<i>Creating a list when elements are known</i> .....	217
<i>Creating a list with dynamic input</i> .....	217
<i>List creation using list() function</i> .....	218
<i>List creation using split() function</i> .....	218
<i>Lists versus immutability</i> .....	219
<i>Accessing elements of list</i> .....	219
<i>By using index</i> .....	219
<i>By using index and for loop</i> .....	221
<i>By using Index and while loop</i> .....	221
<i>By using list slicing</i> .....	222
<i>List comprehension</i> .....	223
<i>List comprehension with for loop</i> .....	224
<i>List comprehension with for loop and if statement</i> .....	225
<i>List comprehension with for loop and nested if statement</i> .....	225
<i>List comprehension with if else statement and for loop</i> .....	226
<i>Nested list comprehension with for loop</i> .....	227
Tuple data structure.....	228

---

<i>Tuple creation</i> .....	229
<i>An empty tuple creation</i> .....	229
<i>Single valued tuple creation</i> .....	230
<i>Multiple valued tuple creation</i> .....	231
<i>Using tuple() function</i> .....	231
<i>Accessing elements of tuple</i> .....	232
<i>By using index</i> .....	232
<i>By using index and for loop</i> .....	233
<i>By using index and while loop</i> .....	234
<i>By using tuple slicing</i> .....	235
<i>Tuple versus immutability</i> .....	236
<i>Tuple comprehension</i> .....	236
<i>List versus tuple comparison</i> .....	237
Set data structure.....	238
<i>Set creation</i> .....	239
<i>Set comprehension</i> .....	240
<i>Set comprehension with for loop</i> .....	241
<i>Set comprehension with for loop and if statement</i> .....	241
<i>Set comprehension with for loop and nested if statement</i> .....	242
<i>Set comprehension with if else statement and for loop</i> .....	243
Dictionary data structure.....	244
<i>Creation of an empty dictionary</i> .....	245
<i>By using dict() function</i> .....	245
<i>By using curly braces only</i> .....	245
<i>Creation of a dictionary</i> .....	246
<i>Accessing dictionary</i> .....	247
<i>Accessing dictionary</i> .....	249
<i>Deleting dictionary item</i> .....	250
<i>Dictionary comprehension</i> .....	252
<i>Dictionary comprehension with for loop</i> .....	252
<i>Dictionary comprehension with for loop and if statement</i> .....	253
<i>Dictionary comprehension with for loop and nested if statement</i> .....	254
<i>Dictionary comprehension with if else statement and for loop</i> .....	255

Conclusion.....	256
Points of remember .....	256
Questions (Long/Short/MCQs).....	257
<b>8. Concept of Packages in Python .....</b>	<b>259</b>
Introduction .....	259
Structure .....	259
Objectives .....	260
Packages.....	260
Structure for package of games.....	262
<i>Using “import” in Packages.....</i>	<i>262</i>
<i>Accessing objects like variables, functions, classes, and lists .....</i>	<i>263</i>
<i>Using “from import” in Packages.....</i>	<i>263</i>
<i>Accessing objects like variables, functions, classes, and lists .....</i>	<i>264</i>
<i>Using “from import *” in Packages.....</i>	<i>264</i>
Accessing games package using different approaches .....	266
Conclusion.....	287
Points to remember .....	287
Questions .....	287
<b>9. Numpy Introduction.....</b>	<b>289</b>
Introduction .....	289
Structure .....	289
Objectives .....	290
Similarities between list and numpy array .....	290
Differences between list and numpy array .....	290
Numpy arrays creation.....	292
<i>1-D array creation using list .....</i>	<i>292</i>
<i>1-D array creation using tuple.....</i>	<i>293</i>
<i>2-D array creation using Nested lists.....</i>	<i>294</i>
<i>Array creation with a particular dtype .....</i>	<i>295</i>
<i>Object type array creation.....</i>	<i>296</i>
<i>1-D array creation with arange() function.....</i>	<i>297</i>
<i>Using linspace() .....</i>	<i>299</i>
<i>Using zeros() .....</i>	<i>301</i>
<i>Using ones() .....</i>	<i>303</i>

---

Using <i>full()</i> .....	304
Using <i>eye()</i> .....	305
Using <i>diag()</i> .....	307
Using <i>empty()</i> .....	309
Comparison between zeros and empty.....	310
Conclusion.....	311
Points to remember.....	311
Questions.....	312
<b>10. Data Visualization Introduction .....</b>	<b>313</b>
Introduction.....	313
Structure.....	314
Objectives.....	314
Python data visualization tools.....	314
Line plot creation by passing 2 ndarrays.....	315
Adding title, xlabel and ylabel to the line plot.....	317
Advanced line plot.....	318
linestyle property.....	320
color property.....	322
default color.....	323
Peep in a shortcut way to set color, marker and linestyle.....	325
<i>mlc form</i> .....	326
<i>clm form</i> .....	327
If no color is mentioned, then default color is blue.....	328
alpha property.....	329
linewidth and markersize property.....	330
markerfacecolor property.....	331
Customizing the figure size.....	333
Plotting multiple lines in a same plot.....	335
Conclusion.....	337
Points of remember.....	337
Questions.....	338
<b>11. Pandas Introduction .....</b>	<b>339</b>
Introduction.....	339
Structure.....	340

---

Objectives .....	340
Pandas Series .....	340
<i>Pandas Series constructor</i> .....	340
<i>Creating Pandas Series by passing a list</i> .....	341
<i>Creating Pandas Series by passing a dictionary</i> .....	342
<i>Creating Pandas Series by passing a numpy array</i> .....	342
<i>Accessing elements in Pandas Series</i> .....	343
<i>Pandas Series slicing</i> .....	344
<i>Pandas Series filtering</i> .....	345
<i>Usage of apply method to Pandas Series</i> .....	347
<i>Aggregating of Pandas Series</i> .....	348
Pandas DataFrame.....	349
<i>Pandas DataFrame constructor</i> .....	350
<i>Pandas DataFrame creation</i> .....	350
<i>Accessing data in Pandas DataFrame</i> .....	351
<i>Data modification in Pandas DataFrame</i> .....	353
<i>Data aggregation in Pandas DataFrame</i> .....	356
Conclusion.....	358
Points to remember .....	358
Questions .....	359
<b>Index</b> .....	<b>359-367</b>



# CHAPTER 1

# Basic Python

# Introduction

## Introduction

In the early 1990s, the Python programming language was created by Guido Van Rossum. Its implementation began in December 1989, in the National Research Institute, Netherlands. The Python language is even older than the Java language. The name Python was coined from one of the most popular British sketch comedy series “*Monty Python’s Flying Circus*”. The first Python version 0.9.0 was released in February, 1991. The next Python version 1.0 was released in January, 1994. Python version 2.0 was released in the October 2000, and version 3.0 came in December 2008. The latest version at the moment, is 3.11.2, which was released in February, 2023.

## Structure

In this chapter, we will discuss the following topics:

- Benefits of Python
- Uses of Python
- Limitations of Python
- Keywords and reserved words

- Identifiers
- Line joining methods
- Print function
- Variables
- Importance of mnemonic variable names in Python
- Concept of Immutability vs Fundamental data types

## Objectives

By the end of this chapter, the reader will learn about the Python language and its benefits, limitations and applications. The rules regarding variables and identifiers will be thoroughly explained with examples. It is important to understand different case styles of the `print()` function, so that the reader can create various Python applications while using it smartly in their code. Finally, the immutability concept with respect to fundamental data types will be elucidated to the reader with examples, for better understanding. Step by step code explanation is also illustrated while explaining various concepts in Python, within the chapter.

## Benefits of Python

It is important to understand why we need to study Python, as it comprises of various benefits:

- Python is a freeware and an open-source beginner's language for newcomers. If anyone is a beginner, Python is the best language to start with. For Java Commercial, the business organization is Oracle; for C#.net, the business organization is Microsoft, but for Python, the business organization is Python Software Foundation. It is a non-profit organization that has IP rights for Python programming language. The website is <https://www.python.org>. Moreover, the source code is open. Based on our requirement, we can customize the Python requirement itself. To work with Java applications, we are required to go for Jython, and for C#.net, we need Iron Python. Similarly, to work with large data (in Machine Learning, Deep Learning), we require Anaconda Python, and to work with Ruby Applications, we require Ruby Python.
- Python is very simple and easy to understand as it is a high-level language. Its syntax is also very easy to comprehend. We do not need to worry for low level activities such as memory management, providing free space and so on. Internally, **Python Virtual Machine (PVM)** will take care.



- Python is platform independent. We can run this language on different hardware platforms such as Linux, Mac, Windows, Raspberry Pi and so on. You can write the program once and run anywhere: this is the concept of platform independent nature. PVM is responsible for running the Python code in different platforms. PVM is platform dependent, whereas Python is platform independent.
- In Python, code can be executed as soon as it is written. Thus, it runs on an interpreter system. It is not required to compile explicitly.
- A programmer can write a program in fewer lines than other programming languages. Thus, it is a more efficient and concise way of writing the code.
- Python can be used as functional oriented, object oriented or procedure oriented.
- Python has tons of libraries which are used for various implementations such as NumPy, SciPy, Pandas and so on.
- Python is extensible. We can use legacy non Python code in our application and fill up the performance gap with other language code.
- Python is a portable language. The Python application can be migrated from one platform to another very easily.
- Python is embedded. We can use the Python script inside Java or C#.Net application. The scope of the Python code is improved. Thus, our application will become scalable.
- It is dynamically typed. We do not need to declare type explicitly. Based on our provided value, the type will be considered automatically, thus giving more flexibility to the programmer. Refer to the following code:

```
a=10
print(type(a))
a='Python'
print(type(a))
a=False
print(type(a))
```

**Output:**

```
<class 'int'>
<class 'str'>
<class 'bool'>
```

**Note:** The preceding code is covered in (Program Name: Chap01\_prog1\_dynamictyped.py)

Python language has similarity to the English language and was designed for readability. The command can be completed with a new line, unlike semicolons or parenthesis, as opposed to other programming languages. To define the scope of loop, classes and functions, Python relies heavily on indentation such as whitespace, unlike other programming languages. which use curly space for this purpose.

## Uses of Python

The Python language is used for:

- Back end web development.
- To create Artificial Intelligence and scientific computing, Machine Learning, Deep Learning, Internet of Things and so on.
- Desktop Applications, 3D graphics, Graphical User Interface Applications.
- Data Analysis, Network applications such as Client Server, Chatting and so on.
- For connectivity to database systems; it can read, write, delete or update the data as per need.
- It can perform very complex mathematics and can handle big data.

A good Python developer writes effective code for backend components, testing and debugging programs. A good developer creates applications that can be integrated with the present ones. Python is used by different companies such as Google, Facebook, Yahoo, NASA, DropBox, BitTorrent, Netflix, YouTube and so on. In Python, most of the syntax has been borrowed from C language and ABC language.

## Limitations of Python

Apart from the benefits, Python also has the following limitations:

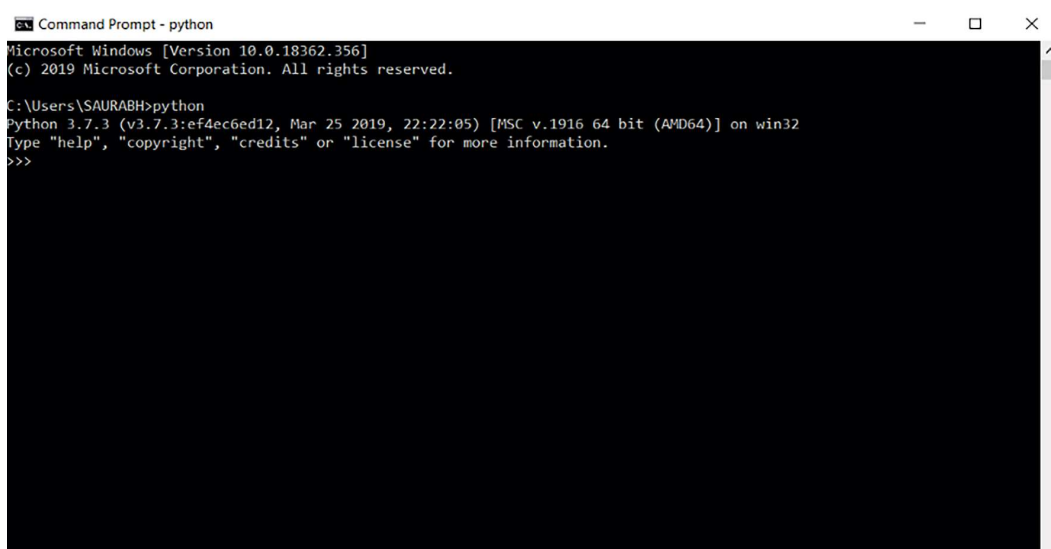
- Python is not suitable for developing mobile applications. This is because currently, it does not have library support to develop mobile applications.
- Python is not the best choice to develop end to end enterprise applications such as banking, telecom applications and so on, as there is no library support.
- The performance is low because the execution is happening line by line. Thus, JIT compiler is added to Python Virtual Machine, so that a group of lines will

be interpreted only once, and each time, the interpreted code is used directly. The preceding flavor is called **PyPy version** (Python for speed).

The popularity of Python in 2023 is record breaking. According to **Stack-Overflow**, Python is the most questioned emerging language, and it is far ahead of competitor languages such as JavaScript, C# and so on. GitHub grants Python the top slot of being the most popular language.

To install python on the system, go to the website <https://www.python.org>. Then go to the **Download** section to download and install the latest Python version (3.11.2, as on today). We have installed Python version on Windows operating system. Keep in mind that we will be doing all our programs in Windows OS only.

It is better to learn Python version 3 instead of version 2 because as of today, all the multinational companies who have been using Python 2, have migrated to Python 3. Python 3.x is developed as a completely independent language and is not an extension of 2.x version. The backward compatibility of Python 3.x is not there for Python 2.x, as there is no guarantee that it will support the same. Moreover, Python 2 may become obsolete in the near future as the libraries will not be maintained. During the installation, an important point to be noted is to tick the checkbox **Add Python 3.7 to Path** (we have installed Python 3.7.3 version in our case). Otherwise, there are chances of error once we try to install our own libraries. Install the setup and you are good to go. Once Python is installed, type **cmd** and enter the word Python. You will get a screen as shown in *Figure 1.1*. In this book, we will learn about Python 3.x and not Python 2.x:

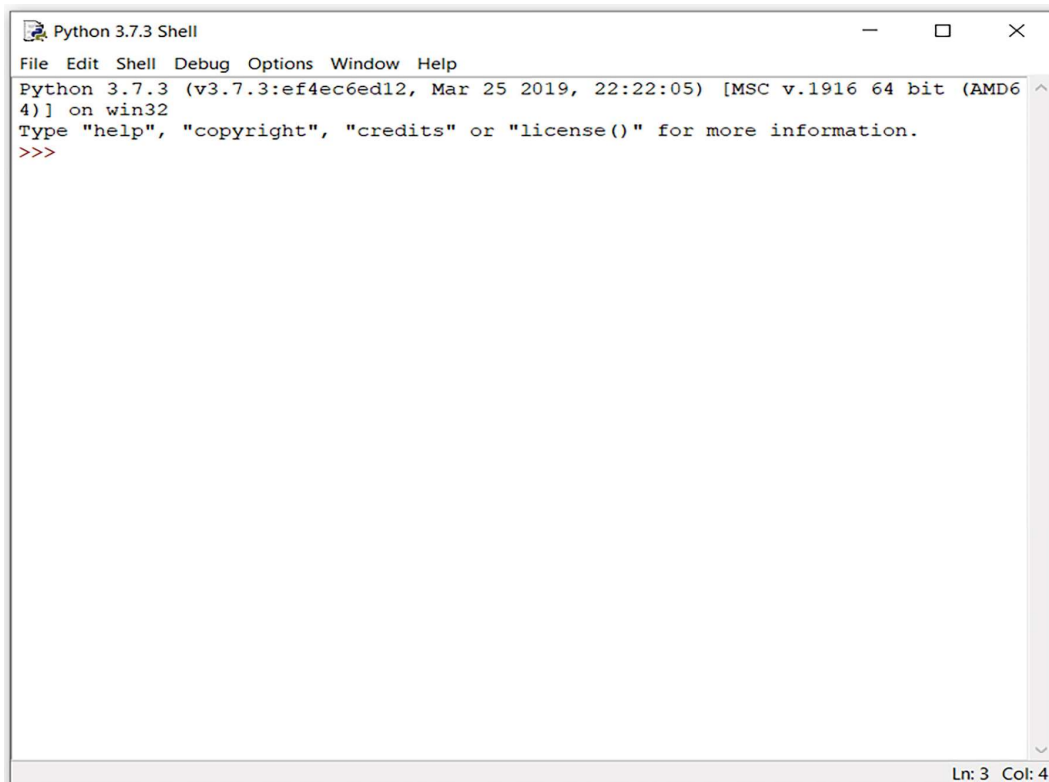


```
Command Prompt - python
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\SAURABH>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

*Figure 1.1: Python version*

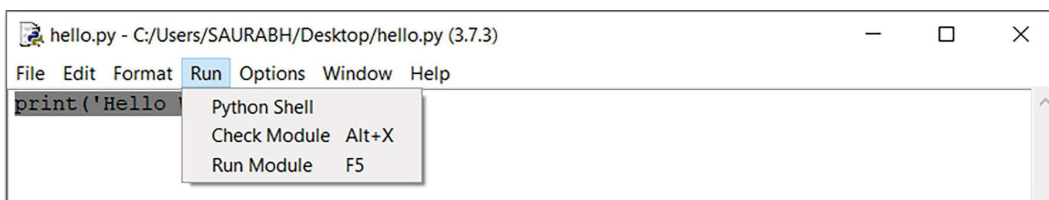
Once we have installed Python, we also get **Integrated Development and Learning Environment (IDLE)** for Python. Here, we can do our coding. It looks as shown in *Figure 1.2*:



*Figure 1.2: Python 3.7.3 shell*

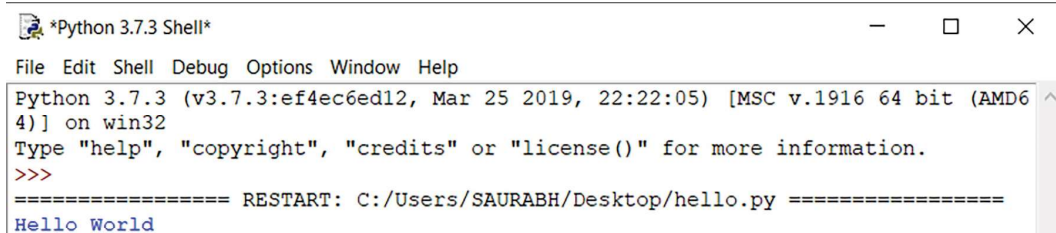
To run the program, follow the given steps:

1. Go to **File | New File**.
2. Type `print('Hello World')` in the file and save it into a respective folder.
3. Click **Run | Run Module F5**, as shown in *Figure 1.3*:



*Figure 1.3: Python file saved as hello.py*

The output **Hello World** will be printed in the IDLE screen, as shown in *Figure 1.4*:



```

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/SAURABH/Desktop/hello.py =====
Hello World

```

*Figure 1.4: Output in Python 3.7.3. shell*

You can also look for other IDEs such as PyCharm, Jupiter and so on. However, we have used **Visual Studio Code (VsCode)**, and integrated gitbash into VSCode. VSCode is a source code editor for making programs which are developed by Microsoft. It has support for debugging, syntax highlighting, code refracting, Intelligent code completion and so on. Before starting with the basics of Python, let us have a brief overview of how to use command line in VSCode.

With the help of command line, we can use files/folders, quickly create or remove them, as well as copy and move the files to/from the folder, among others. Let us start by typing the following commands in the bash terminal of VSCode, and thus visualize the output. The commands are typed in bold letters.

- **pwd**: The **pwd** command or the **Print working directory** will display the current location of the working directory. From the terminal output shown in *Figure 1.5*, we can see that the working directory is **E:/python\_progs**.



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/python_progs
$ pwd
/e/python_progs

```

*Figure 1.5: pwd command*

- **ls**: This command will display the list of files and folders in the terminal. From the terminal output shown in *Figure 1.6*, we can see the list of files and directories within the file system:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/python_progs
$ ls
activationcode.txt      prog19_functions.py      prog42_iterablevsiterator.py  prog72_errorsraise.py
circuit-VRC1-vc.png    prog2_escapesesequence.py  prog43_zip.py                 prog73_exceptionhandling.py
copyingsalary.txt      prog20_fibonacci.py       prog49_practicequestions.py  prog74_debugging.py
demo_binpy1.py         prog21_defaultargs_func.py  prog5_calculations.py        prog75_readtextfiles.py
demofolder             prog22_scope.py           prog50_any_all.py            prog76_readfromonefile_copy.py
editimages.py         prog23_list.py            prog51_advancedadmin_max.py  prog77_htmlread_link.py
emojis.txt             prog24_listsvsarray.py     prog52_advancedsort.py      prog78_readcsv.py
file.csv               prog25_loop_in_list.py     prog53_docstrings.py        prog79_writecsv.py
file1.txt              prog26_listinsidelist.py   prog54_firstclosurefunc.py  prog80_userinput.py
file2.txt              prog27_moreaboutlists.py   prog55_function_argument.py  prog80_read_write.py
filename.csv           prog28_function_list_examples.py  prog56_function_returning_function.py  prog81_osimport.py
gui_file1.txt          prog29_tuples.py          prog57_decorators.py        prog82_movingfiles_folder.py
index.html             prog3_escape_asnormaltext.py  prog58_decorators_practice.py  prog9_string_properties.py
mk_file.txt            prog30_dictionary.py       prog59_decorator_with_arguments.py  proggui_1.py
numpy_eg2.py           prog31_add_deletedata_dictionary.py  prog6_variables.py           proggui_2_creatingwidgetsusingloop.py
numpyeg1.py           prog32_sets.py             prog60_generators.py        proggui_3_LabelFrames.py
output.txt             prog33_listcomprehension.py  prog61_oops.py              PROGGUI_4_TABBEDCONTROL.PY
prog1_printeg.py      prog34_dictionarycomprehension.py  prog62_classvariable.py     proggui_6_msgbox_exceptionhandling.py
prog10_string_methods.py  prog35_setscomprehension.py  prog63_classmethods.py      progui_5_menubar.py
prog11_stripmethod.py  prog36_args.py             prog64_classmethodsasconstructor.py  python_syllabi.txt
```

Figure 1.6: `ls` command

- **ls -l:** It will display the list of files and folders in the terminal, using a long listing format. From the terminal output shown in Figure 1.7, we can see the list of files and directories within the file system, along with the owner, permissions. Since the list of contents is so long, only a part of it has been shown.

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/python_progs
$ ls -l
total 412
-rw-r--r-- 1 SAURABH 197121 100 Aug 20 19:21 activationcode.txt
-rw-r--r-- 1 SAURABH 197121 23060 Aug 18 13:12 circuit-VRC1-vc.png
-rw-r--r-- 1 SAURABH 197121 113 Aug 28 07:35 copyingsalary.txt
-rw-r--r-- 1 SAURABH 197121 996 Aug 21 19:23 demo_binpy1.py
drwxr-xr-x 1 SAURABH 197121 0 Aug 29 07:32 demofolder
-rw-r--r-- 1 SAURABH 197121 2112 Aug 29 22:05 editimages.py
-rw-r--r-- 1 SAURABH 197121 83941 Aug 28 19:51 emojis.txt
-rw-r--r-- 1 SAURABH 197121 110 Aug 31 19:03 file.csv
-rw-r--r-- 1 SAURABH 197121 24 Aug 28 07:28 file1.txt
-rw-r--r-- 1 SAURABH 197121 24 Aug 28 07:28 file2.txt
-rw-r--r-- 1 SAURABH 197121 104 Aug 28 21:54 filename.csv
-rw-r--r-- 1 SAURABH 197121 277 Aug 31 18:13 gui_file1.txt
-rw-r--r-- 1 SAURABH 197121 672 Aug 28 19:32 index.html
-rw-r--r-- 1 SAURABH 197121 0 Aug 29 06:59 mk_file.txt
-rw-r--r-- 1 SAURABH 197121 5726 Sep 3 20:52 numpy_eg2.py
-rw-r--r-- 1 SAURABH 197121 2140 Sep 1 16:22 numpyeg1.py
-rw-r--r-- 1 SAURABH 197121 65 Aug 28 19:39 output.txt
-rw-r--r-- 1 SAURABH 197121 435 Aug 14 18:04 prog1_printeg.py
```

Figure 1.7: `ls -l` command

- **clear:** This command will clear the screen terminal. From the terminal output shown in Figure 1.8, we can see that the screen has been cleared:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/python_progs
$
```

Figure 1.8: `clear` command

- **cd**: It will change the current working directory in the operating systems. From the terminal output shown in *Figure 1.9*, we can see that the directory has been changed to **E://democreated**:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/python_progs
$ cd E://democreated

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ pwd
/e/democreated
```

*Figure 1.9: cd command*

- **mkdir**: This command is used to create folders in the operating system. From the terminal output shown in *Figure 1.10*, we can see that a new folder namely **command folder**, has been created:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ mkdir commandfolder

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ ls
commandfolder
```

*Figure 1.10: mkdir command*

An important point to observe is that if we give space inbetween the folder name, and the folder name is not present in double inverted commas, then separate individual folders will be created. In *Figure 1.11*, we can see that 3 different folders are created from **mkdir command practice 2**:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ mkdir command practice 2

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ ls
2 command commandfolder practice
```

*Figure 1.11: mkdir without double inverted commas*

Now, let us put double inverted comma between the folder name: **mkdir "command practice 2"**. From *Figure 1.12*, we can see that a new folder **command practice 2** is created:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ mkdir "command practice 2"

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ ls
2  command 'command practice 2'  commandfolder  practice
```

Figure 1.12: `mkdir` with double inverted commas

Let us assume we need to move to a folder **command practice 2**. Type the command `cd "command practice 2 "/"`, as shown in Figure 1.13:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ cd "command practice 2"/

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/command practice 2
$ pwd
/e/democreated/command practice 2
```

Figure 1.13: Moving to a folder

- **touch**: This command is used to set the modification and access times of files to the current time of day. If the file is not present, then it will create the file with default permissions. In Figure 1.14, a text file namely **newfile.txt** is created:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/command practice 2
$ touch newfile.txt

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/command practice 2
$ ls
newfile.txt
```

Figure 1.14: `touch` (text file)

We can even create a Python file. A new file namely **demo.py** is created, in Figure 1.15:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/command practice 2
$ touch demo.py

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/command practice 2
$ ls
demo.py  newfile.txt
```

Figure 1.15: `touch` (python file)



- **rm**: This command is used to remove objects such as file, directories and so on. In *Figure 1.16*, let us remove **newfile.txt**:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/command practice 2
$ rm newfile.txt

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/command practice 2
$ ls
demo.py
```

*Figure 1.16: remove a text file*

- **cd ..**: This command is used to move one folder back. in *Figure 1.17*, we can see that one folder has been moved back:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/command practice 2
$ cd ..

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ pwd
/e/democreated
```

*Figure 1.17: command to move one folder back*

- **rm -rf**: This command is used to remove folders completely. **rf** stands for recursive force. In *Figure 1.18*, we can see that the folder **command practice 2** has been removed completely:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ rm -rf "command practice 2/"

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated
$ ls
2 command commandfolder practice
```

*Figure 1.18: rm -rf command*

- **mv**: This command is used to rename a file with a new name or move a file name into a new folder. In *Figure 1.19*, we can see that the file name **file1.py** has been renamed to **file.py** from the current folder **cd commandfolder/**:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ ls
file1.py

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ mv file1.py file.py

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ ls
file.py
```

*Figure 1.19: mv command (rename)*

In *Figure 1.20*, we can see that the file name `file.py` has been moved to the current folder `mvfolder`.

Firstly, we have created a folder named `mvfolder`. Then, we have moved the file into that folder using command `mv file.py ./mvfolder/`

On the current path, we typed `ls` to see the list of files and folders. We can see that the file has moved into the `mvfolder` since only the folder name is being displayed. We then change the directory into `mvfolder`. We typed `ls` to see the list of files and folders.

Refer to *Figure 1.20*:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ mkdir mvfolder

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ ls
file.py  mvfolder

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ mv file.py ./mvfolder/

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ ls
mvfolder

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ cd mvfolder/

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder/mvfolder
$ ls
file.py
```

*Figure 1.20: mv command (move file into folder)*

Let us now assume that we again want to move the file one folder backwards. In this case, we have to use filename with “`..`”, as shown in *Figure 1.21*. After typing the command `mv file.py ..`, the file `file.py` has moved to the folder `commandfolder` as shown:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder/mvfolder
$ mv file.py ..

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder/mvfolder
$ ls

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder/mvfolder
$ cd ..

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ ls
file.py  mvfolder
```

*Figure 1.21: mv command (move file into one folder backward)*

Let us now assume that we want to copy a file into the folder **mvfolder**. In *Figure 1.22*, we can see that the command `cp file.py` has been copied into the folder `/mvfolder/`. We can see that using `ls` command, the folder **mvfolder** contains the file name `file.py`:

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ cp file.py ./mvfolder/

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ ls
file.py mvfolder

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ cd mvfolder/

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder/mvfolder
$ ls
file.py
```

*Figure 1.22: cp command*

If you want to save the list of commands, type in the bash terminal of VsCode into a text file, and then use the following command:

```
history > history_for_print.txt
```

Here, **history** is the command and **history\_for\_print.txt** is the text filename as shown in *Figure 1.23*. The command and the file name is joined by the `>` symbol. We can see that the list of commands typed has been saved in the text file name **history\_for\_print.txt**.

```
SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ history > history_for_print.txt

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ ls
file.py history_for_print.txt mvfolder

SAURABH@LAPTOP-NHFM79LF MINGW64 /e/democreated/commandfolder
$ code history_for_print.txt
```

*Figure 1.23: history command*

## Keywords and reserved words

In any language, whether it may be a general speaking language such as English, or a programming language such as Python or C, there are some reserved words

to represent some meaning or functionality, and they are called reserved words or keywords. There are tons of English reserved words, each with some specific meaning. It is quite impossible to remember those words. On the other hand, if we look at some programming languages like Java, only 53 reserved words are present. However, in Python, there are only 35 reserved words.

Thus, if we understand these 35 keywords, we might become an expert on this language. To know the total keywords in the Python language, type the following command in VsCode:

```
import keyword
print(keyword.kwlist)
```

*Note:* The preceding code is covered in (Program Name: Chap01\_prog2\_keywords.py)

Save this to a file named as **python\_keywords.py**. All reserved words in Python contain alphabet symbols. To run the above program, type the command as:

```
python python_keywords.py.
```

You will get the list of keywords as shown:

```
['False', 'None', 'True',
'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue',
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with', 'yield']
```

We will discuss all the list of keywords as per requirement. However, the important observation that you may get by looking into the keywords is as follows:

- All the 35 keywords contain only alphabets.
- True, False and None are the only 3 keywords which are in Uppercase. The remaining 32 keywords are in lower case.
- An important point to note is that there is no **switch** or **do-while** concepts in Python.
- Since Python is dynamically typed, there is no reserved words such as int, float, Boolean, complex data types and so on.

To check if any word is a keyword or not, type the following command:

```
print(keyword.iskeyword('yield'))
```

On running the preceding file, you will get a Boolean value as True, indicating that the yield word is a keyword. You can continue to check with other reserved words. The return type of `iskeyword` is either True or False.

## Identifiers

Identifiers are nothing but the user defined names utilized in the programs, to represent a variable, a function, a class or a module. Identifiers can contain a letter, digit or underscore. The first letter of the identifier must be a letter or underscore, and should not start with digit. An example of an identifier is as follows:

- `var_1=67`: Variable name is an example of an identifier.
- `_var = 4`: The first letter of an identifier can be an underscore.
- `3sd = 7`: Invalid. `SyntaxError: invalid syntax`.
- Special characters are excluded from the identifier.
- `var@2 = 8`: `SyntaxError: cannot assign to operator`.

The allowed characters in Python are alphabets (either upper or lower case), digits (from 0 to 9) and the underscore symbol. As the Python language is case sensitive, identifiers themselves are case sensitive. The variables, for example, `sum` and `SUM` are different. If the identifier starts with underscore, then it is private. Reserved words are not allowed in identifiers. There is no length limit for identifiers, but care has to be taken that it is not too lengthy and must be meaningful enough for even a new person to understand.

## Line joining methods

Whenever we are trying to divide input into different physical line, there will be an error. For example:

```
print("hello
```

When we try to write the rest of the words in the next line, the following error will be displayed:

```
    print("hello
          ^
```

```
SyntaxError: EOL while scanning string literal
```

This can be overcome by using implicit and explicit line joining method.

## Implicit line joining method

In implicit line joining method, the expression is split into multiple lines using parenthesis, curly lines or square brackets, as discussed.

### Using curly braces

Refer to the expression:

```
days = {'Mon', 'Tue', 'Wed',  
        'Thu', 'Fri', 'Sat',  
        'Sun'}  
  
print(days)
```

#### Output

```
{'Tue', 'Sat', 'Fri', 'Sun', 'Mon', 'Wed', 'Thu'}
```

*Note:* You may get different output every time here.

### Using square brackets

Refer to the expression:

```
days = ['Mon', 'Tue', 'Wed',  
        'Thu', 'Fri', 'Sat',  
        'Sun']  
  
print(days)
```

#### Output:

```
['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
```

### Using parenthesis

Refer to the expression:

```
days = ('Mon', 'Tue', 'Wed',  
        'Thu', 'Fri', 'Sat',  
        'Sun')
```

```
print(days)
```

**Output:**

```
('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')
```

The preceding examples can also be written with comments and blank lines, as shown:

- We can write comment after the line:

```
days = {'Mon','Tue','Wed', # Mon - Wed
        'Thu','Fri','Sat', # Thu - Sat
        'Sun'} # sun
```

```
print(days)
```

**Output**

```
{'Tue', 'Thu', 'Fri', 'Sat', 'Sun', 'Mon', 'Wed'}
```

- Blank lines can be inserted:

```
days = { #Blank
        'Mon','Tue','Wed', # Mon - Wed
        'Thu','Fri','Sat', # Thu - Sat
        'Sun'} # sun
```

```
print(days)
```

**Output**

```
{'Sun', 'Tue', 'Thu', 'Sat', 'Wed', 'Mon', 'Fri'}
```

*Note:* All the implicit joining method examples are covered in (Program Name: Chap01\_prog3\_implicit\_joining\_method.py)

## Explicit joining method

In the explicit line joining method, we are using backward slash to split the input. The backward slash is used to join the logical lines together in such a manner that they are declared in a single line. Refer to the following examples.

### Example 1

Refer to the following expression:

```
print("hello\
```