4th Edition

YASHAVANT KANETKAR
ADITYA KANETKAR

Python Is Future, Embrace It Fast
Learn Python Quickly
A Programmer-Friendly Guide

Let Us Python

bpb

Brief Contents

1	Introduction to Python	1
2	Getting Started	9
3	Python Basics	19
4	Strings and Regular Expressions	41
5	Decision Control Instruction	63
6	Repetition Control Instruction	77
7	Console Input/Output	89
8	Lists	99
9	Tuples	117
10	Sets	129
11	Dictionaries	139
12	Comprehensions	153
13	Functions	165
14	Recursion	183
15	Functional Programming	199
16	Modules and Packages	213
17	Namespaces	225
18	Classes and Objects	235
19	Intricacies of Classes and Objects	253
20	Containership and Inheritance	267
21	Iterators and Generators	285
22	Exception Handling	307
23	File Input/Output	325
24	Miscellany	341
25	Concurrency and Parallelism	363
26	Synchronization	381

facebook.comRakeshPrasadGupta2024

27 Numpy Library 403
Appendix A - Precedence Table 419
Appendix B - Debugging in Python 421
Appendix C - Chasing the Bugs 427
Periodic Tests 433
Index 447

facebook.comRakeshPrasadGupta2024

Contents

1 **Introduction to Python** 1

What is Python? 2
Reasons for Popularity 2
What sets Python apart? 3
Where is Python used? 3
Who uses Python today? 4
Programming Paradigms 4
Functional Programming Model 5
Procedural Programming Model 5
Object-oriented Programming Model 6
Event-driven Programming Model 6
Exercises 7

2 **Getting Started** 9

Python Specification 10
Python Installation under Windows 10
Python Installation under Linux 11
Python Resources 12
Third-party Packages 12
More Sophisticated Tools 13
Compilation Approach in C, C++ 13
Compilation Approach in Python 14
Pros and Cons of Traditional Approach 15
Pros and Cons of Modern Approach 15
Working with Python 16
Python Programming Modes 16
Determining Python Version 17
Exercises 18

3 **Python Basics** 19

Identifiers and Keywords 20
Python Types 20
Basic Types 21
Integer and Float Ranges 21
Variable Type and Assignment 22
Arithmetic Operators 22
Operation Nuances 23
Precedence and Associativity 24
Conversions 25
Built-in Functions 25

facebook.comRakeshPrasadGupta2024

Built-in Modules 26
Container Types 28
Python Type Jargon 28
Comments and Indentation 29
Multi-lining 29
Classes and Objects 30
Multiple Objects 31
Programs 32
Exercises 39

4 Strings and Regular Expressions 41
What are Strings? 42
Accessing String Elements 42
String Properties 43
Built-in Functions 44
String Methods 44
String Conversions 45
String Comparisons 46
Byte Sequences 46
Regular Expressions 47
Regex Functionality 48
Regex Metacharacters 48
Regex Repetition Qualifiers 50
Regex Anchors 51
Regex Grouping 51
Programs 51
Exercises 59

5 Decision Control Instruction 63
Decision Control Instruction 64
Nuances of Conditions 65
Logical Operators 65
Conditional Expressions 66
all( ) and *any( )* 67
Receiving Input 67
pass Statement 68
Programs 68
Exercises 73

6 Repetition Control Instruction 77
Repetition Control Instruction 78
Usage of *while* Loop 78
Usage of *for* Loop 79
break and *continue* 81

	Else Block of a Loop	81
	Programs	82
	Exercises	85
7	**Console Input/Output**	**89**
	Console Input	90
	Console Output	91
	Formatted Printing	91
	Programs	93
	Exercises	96
8	**Lists**	**99**
	What are Lists?	100
	Accessing List Elements	100
	Looping in Lists	101
	Basic List Operations	101
	Using Built-in Functions on Lists	104
	List Methods	105
	Sorting and Reversing	105
	List Varieties	106
	Stack Data Structure	107
	Queue Data Structure	107
	Programs	107
	Exercises	113
9	**Tuples**	**117**
	What are Tuples?	118
	Accessing Tuple Elements	118
	Looping in Tuples	119
	Basic Tuple Operations	119
	Using Built-in Functions on Tuples	120
	Tuple Methods	121
	Tuple Varieties	121
	Programs	122
	Exercises	127
10	**Sets**	**129**
	What are Sets?	130
	Accessing Set Elements	131
	Looping in Sets	131
	Basic Set Operations	131
	Using Built-in Functions on Sets	132
	Set Methods	132
	Mathematical Set Operations	133
	Updating Set Operations	134

facebook.comRakeshPrasadGupta2024

Set Varieties 134
Programs 134
Exercises 137
11 Dictionaries 139
What are Dictionaries? 140
Accessing Dictionary Elements 140
Looping in Dictionaries 141
Basic Dictionary Operations 141
Using Built-in Functions on Dictionaries 142
Dictionary Methods 143
Dictionary Varieties 143
Programs 144
Exercises 149
12 Comprehensions 153
What are Comprehensions? 154
List Comprehension 154
Set Comprehension 155
Dictionary Comprehension 156
Programs 156
Exercises 162
13 Functions 165
What are Functions? 166
Communication with Functions 167
Types of Arguments 168
Unpacking Arguments 171
Programs 172
Exercises 181
14 Recursion 183
Repetitions 184
Recursive Function 184
When to use Recursion 185
Problem as Similar Sub-problems 185
Recursive Factorial Function 186
Problem with Unknown Loops 187
Types of Recursion 189
Recursion Limit 190
Iteration to Recursion 190
Programs 190
Exercises 196
15 Functional Programming 199
Functional Programming 200

facebook.comRakeshPrasadGupta2024

Functions as First-class Values 200
Lambda Functions 201
Higher Order Functions 202
Map, Filter, Reduce 203
map( ) Function 203
filter( ) Function 204
reduce( ) Function 204
Using Lambda with *map( ), filter( ), reduce( )* 205
Where are they Useful? 206
Programs 207
Exercises 211

16 Modules and Packages 213
The Main Module 214
Multiple Modules 214
Importing a Module 215
Variations of *import* 216
Search Sequence 216
Same Code, Different Interpretation 217
Packages 217
Third-party Packages 218
Programs 219
Exercises 223

17 Namespaces 225
Symbol Table 226
Namespace 226
globals( ) and *locals( )* 227
Where to use them? 228
Inner Functions 229
Scope and LEGB Rule 230
Programs 231
Exercises 233

18 Classes and Objects 235
Programming Paradigms 236
What are Classes and Objects? 236
Classes and Objects in Programming 237
User-defined Classes 237
Access Convention 239
Object Initialization 240
Class Variables and Methods 242
vars( ) and *dir( )* Functions 242
More *vars( )* and *dir( )* 247

facebook.comRakeshPrasadGupta2024

Programs 244
Exercises 249

19 Intricacies of Classes and Objects 253
Identifier Naming Convention 254
Calling Functions and Methods 255
Operator Overloading 256
Which Operators to Overload 257
Everything is an Object 288
Imitating a Structure 260
Type Conversion 260
Programs 261
Exercises 264

20 Containership and Inheritance 267
Reuse Mechanisms 268
Which to use When? 268
Containership 268
Inheritance 269
What is Accessible where? 271
isinstance( ) and *issubclass( )* 272
The *object* Class 273
Features of Inheritance 273
Types of Inheritance 273
Diamond Problem 275
Abstract Classes 276
Runtime Polymorphism 277
Programs 277
Exercises 282

21 Iterators and Generators 285
Iterables and Iterators 286
zip( ) Function 286
Iterators 287
User-defined Iterators 289
Generators 290
Which to use When? 291
Generator Expressions 291
Programs 292
Exercises 305

22 Exception Handling 307
What may go Wrong? 308
Syntax Errors 308
Exceptions 309

facebook.comRakeshPrasadGupta2024

How to deal with Exceptions? 310
How to use *try* - *except*? 311
Nuances of *try* and *except* 312
User-defined Exceptions 313
else Block 316
finally Block 317
Exception Handling Tips 317
Programs 317
Exercises 322

23 File Input/Output **325**
I/O System 326
File I/O 326
Read / Write Operations 327
File Opening Modes 328
with Keyword 329
Moving within a File 329
Serialization and Deserialization 330
Serialization of User-defined Types 333
File and Directory Operations 334
Programs 335
Exercises 338

24 Miscellany **341**
Documentation Strings 342
Command-line Arguments 343
Parsing of Command-line 344
Bitwise Operators 346
Assertion 347
Decorators 348
Decorating Functions with Arguments 350
Unicode 353
bytes Datatype 353
Create Executable File 355
Programs 356
Exercises 361

25 Concurrency and Parallelism **363**
Concurrency and Parallelism 364
What are Threads? 364
Concurrency and Parallelism in Programming 365
CPU-bound and I/O-bound Programs 366
Which to use When? 366
Concurrency for Improving Performance 362

facebook.comRakeshPrasadGupta2024

Types of Concurrencies 368
Thread Properties 368
Launching Threads 369
Passing Parameters to a Thread 370
Programs 370
Exercises 378

26 Synchronization 381

Synchronization 382
Examples of Sharing Resources 382
Example of Communication between Threads 383
Mechanisms for Sharing Resources 383
Lock 383
RLock 384
Semaphore 388
Mechanisms for Inter-thread Communication 388
Event 386
Condition 386
Programs 387
Exercises 401

27 Numpy Library 403

Creation of Array 404
Creation of Filler Arrays 405
Array Attributes 406
Array Operations 407
Arithmetic Operations 407
Statistical Operations 408
Linear Algebra Operations 408
Bitwise Operations 409
Copying and Sorting 409
Comparison 410
Indexing and Slicing 411
Array Manipulation 412
Programs 413
Exercises 417

Appendix A - Precedence Table 419

Appendix B - Debugging in Python 421

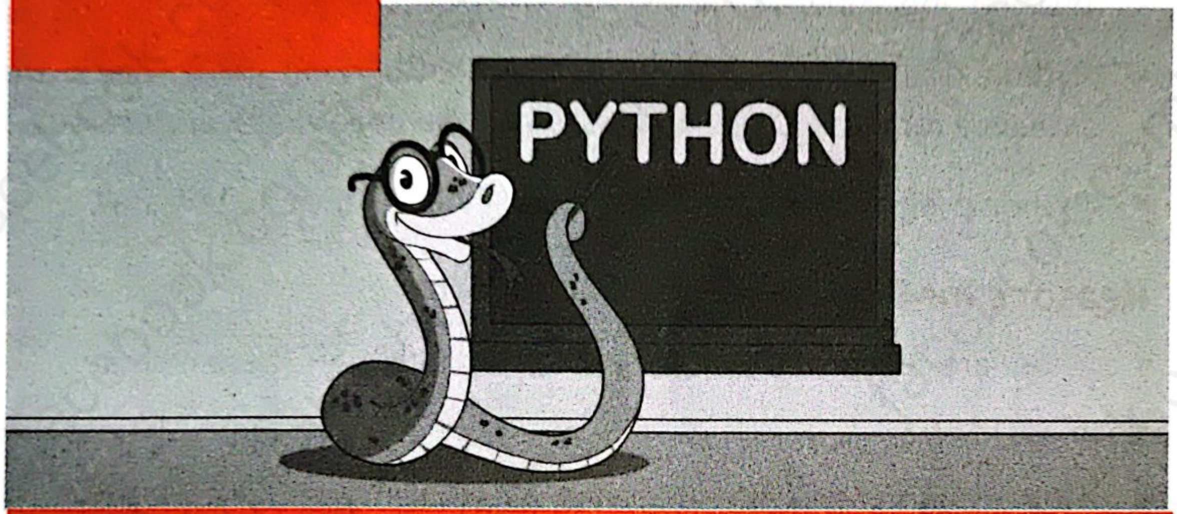
Appendix C - Chasing the Bugs 427

Appendix D - Periodic Tests 433

Index 447

facebook.comRakeshPrasadGupta2024

1 Introduction to Python



"Wet your feet..."

Contents

- What is Python?
- Reasons for Popularity
- What sets Python apart?
- Where is Python Used?
- Who uses Python today?
- Programming Paradigms
- Functional Programming Model
- Procedural Programming Model
- Object-oriented Programming Model
- Event-driven Programming Model
- Exercises

facebook.comRakeshPrasadGupta2024

What is Python?

- Python is a high-level programming language created by Guido Van Rossum - fondly known as Benevolent Dictator For Life.
- Python was first released in 1991. Today Python interpreters are available for many Operating Systems including Windows and Linux.
- Python programmers are often called Pythonists or Pythonistas.

Reasons for Popularity

- There are severa' reasons for Python's popularity. These include:

(a) Free:

- Python is free to use and distribute and is supported by community.
- Python interpreter is available for every major platform.

(b) Software quality:

- Better than traditional and scripting languages.
- Readable code, hence reusable and maintainable.
- Support for advance reuse mechanisms.

(c) Developer productivity:

- Much better than statically typed languages.
- Much smaller code.
- Less to type, debug and maintain.
- No lengthy compile and link steps.

(d) Program portability:

- Python programs run unchanged on most platforms.
- Python runs on every major platform currently in use.
- Porting program to a new platform usually need only cut and paste. This is true even for GUI, DB access, Web programming, OS interfacing, Directory access, etc.

(e) Support libraries:

- Strong library support from Text pattern matching to networking.
- Vast collection of third-party libraries.
- Libraries for Web site construction, Numeric programming, Game development, Machine Learning etc.

facebook.comRakeshPrasadGupta2024

(f) Component integration:

- Can invoke C, C++ libraries and Java components.
- Can communicate with frameworks such as COM, .NET.
- Can interact over networks with interfaces like SOAP, XML-RPC, CORBA.
- With appropriate glue code, Python can subclass C++, Java, C#. classes, thereby extending the reach of the program.
- Popularly used for product customization and extension.

(g) Enjoyment:

- Ease of use.
- Built-in toolset.
- Programming becomes pleasure than work.

What sets Python apart?

(a) Powerful:

- Dynamic typing.
- No variable declaration.
- Automatic allocation and Garbage Collection.
- Supports classes, modules and exceptions.
- Permits componentization and reuse.
- Powerful containers - Lists, Dictionaries, Tuples, etc.

(b) Ready-made stuff:

- Support for operations like joining, slicing, sorting, mapping, etc.
- Powerful library.
- Large collection of third-party utilities.

(c) Ease of use:

- Type and run.
- No compile and link steps.
- Interactive programming experience.
- Rapid turnaround.
- Programs are simpler, smaller and more flexible.

Where is Python used?

- Python is used for multiple purposes. These include:

(a) System programming

(b) Building GUI applications

(c) Internet scripting

facebook.comRakeshPrasadGupta2024

(d) Component integration

(e) Database programming

(f) Rapid prototyping

(g) Numeric and Scientific programming

(h) Game programming

(i) Robotics programming

Who uses Python today?

- Many organizations use Python for varied purposes. These include:

(a) Google - In web search system

(b) YouTube - Video Sharing service

(c) Bit-torrent - Peer to Peer file sharing system

(d) Intel, HP, Seagate, IBM, Qualcomm - Hardware testing

(e) Pixar, Industrial Light and Magic - Movie animation

(f) JP Morgan, Chase, UBS - Financial market forecasting

(g) NASA, FermiLab - Scientific programming

(h) iRobot - Commercial robot vacuum cleaners

(i) NSA - Cryptographic and Intelligence analysis

(j) IronPort - Email Servers

Programming Paradigms

- Paradigm means organization principle. It is also known as model.
- Programming paradigm/model is a style of building the structure and elements of computer programs.
- There exist many programming models like Functional, Procedural, Object-oriented, Event-driven, etc.
- Many languages facilitate programming in one or more paradigms. For example, Python supports Functional, Procedural, Object-oriented and Event-driven programming models.

facebook.comRakeshPrasadGupta2024

- There are situations when Functional programming is the obvious choice, and other situations where Procedural programming is the better choice.
- Paradigms are not meant to be mutually exclusive. A single program may use multiple paradigms.

Example

Functional Programming Model

- Functional programming decomposes a problem into a set of functions. These functions provide the main source of logic in the program.
- Functions take input parameters and produce outputs. Python provides functional programming techniques like lambda, map, reduce and filter. These are discussed in Chapter 15.
- In this model computation is treated as evaluation of mathematical functions. For example, to get factorial value of a number, or nth Fibonacci number we can use the following functions:

```
factorial(n)  = 1                    if n == 0
          = n * factorial(n - 1)    if n > 0

fibo(n) = 0                 if n = 0
        = 1                 if n = 1
        = fibo(n - 2) + fibo(n - 1)     if n > 1
```

- The output value of a function depends only on its arguments, so calling a function with the same value for an argument always produces the same result. As a result, it is a good fit for parallel execution.
- No function can have side effects on other variables (state remains unaltered).
- Functional programming model is often called a 'Declarative' programming paradigm as programming is done with expressions or declarations instead of statements.

Procedural Programming Model

- Procedural programming solves the problem by implementing one statement (a procedure) at a time. Thus, it contains explicit steps that are executed in a specific order.

- It also uses functions, but these are not mathematical functions like the ones used in functional programming. Functional programming focuses on expressions, whereas Procedural programming focuses on statements.
- The statements don't have values and instead modify the state of some conceptual machine.
- Same language expression can result in different values at different times depending on the global state of the executing program. Also, the functions may change a program's state.
- Procedural programming model is often called Necessary 'Imperative' programming as it changes state with an explicit sequence of statements.

Object-oriented Programming Model

- This model mimics the real world by creating inside the computer a mini-world of objects.
- In a University system objects can be VC, Professors, Non-teaching staff, students, courses, semesters, examinations, etc.
- Each object has a state (values) and behavior (interface/methods). Objects get state and behavior based on the class from which it created.
- Objects interact with one another by sending messages to each other, i.e., by calling each other's interface methods.

Event-driven Programming Model

- This model is popularly used for programming GUI applications containing elements like windows, check boxes, buttons, combo-boxes, scroll bars, menus, etc.
- When we interact with these elements (like clicking a button, or moving the scrollbar or selecting a menu item) events occur and these elements emit messages. There are listener methods which are registered with these GUI elements which react to these events.
- Since there is no guaranteed sequence in which events may occur (based on how we interact with GUI elements), the listeners should be able to handle them in asynchronous manner.

EX Exercises

[A] Answer the following:

(a) Mention 5 fields in which Python is popularly used.

(b) Where is event-driven programming popularly used?

(c) Why Python is called portable language?

(d) What is the single most important feature of different programming models discussed in this chapter?

(e) Which of the following is not a feature of Python?

- Static typing
- Dynamic typing
- Run-time error handling through error numbers
- Library support for containers like Lists, Dictionaries, Tuples

(f) Give an example application of each of the following programming models:

- Functional model
- Procedural model
- Object-oriented model
- Event-driven model

[B] State whether the following statements are True or False:

(a) Python is free to use and distribute.

(b) Same Python program can work on different OS - microprocessor combinations.

(c) It is possible to use C++ or Java libraries in a Python program.

(d) In Python type of the variable is decided based on its usage.

(e) Python cannot be used for building GUI applications.

(f) Python supports functional, procedural, object-oriented and event-driven programming models.

(g) GUI applications are based on event-driven programming model.

facebook.comRakeshPrasadGupta2024

(h) Functional programming model consists of interaction of multiple objects.

[C] Match the following pairs:

a. Functional programming	1. GUI element based interaction
b. Event-driven programming	2. Interaction of objects
c. Procedural programming	3. Statements
d. OOP	4. Maths-like functions

[D] Fill in the blanks:

(a) Functional programming paradigm is also known as ________ programming model.

(b) Procedural programming paradigm is also known as ________ programming model.

(c) Python was created by ________.

(d) Python programmers are often called ________.

facebook.comRakeshPrasadGupta2024