

Contents

- 1 The World of Programming 1**
 - 1.1 Programming Languages 5
 - 1.1.1 Low-Level Programming Languages 5
 - 1.1.2 High-Level Programming Languages 6
 - 1.2 Programming Paradigms 7
 - 1.2.1 The Imperative Programming Paradigm 8
 - 1.2.2 The Functional Programming Paradigm 8
 - 1.2.3 The Logical-Declarative Programming Paradigm 9
 - 1.2.4 The Object-Oriented Programming Paradigm 9
 - 1.2.5 The Concurrent Programming Paradigm 11
 - 1.2.6 The Event-Driven Programming Paradigm 11
 - 1.3 The Zoo of Programming Languages 12
 - 1.3.1 How to Choose a Programming Language for an
Implementation 13
 - 1.4 How Programing Languages Are Implemented 16
 - 1.4.1 Compilative Approach 17
 - 1.4.2 Interpretive Approach 19
 - 1.4.3 Mixed Approaches 20
 - 1.5 How a Program Gets “Written” 21
 - 1.5.1 Modular & Functional Break-Down 21
 - 1.5.2 Testing 22
 - 1.5.3 Errors 24
 - 1.5.4 Debugging 27
 - 1.5.5 Good Programming Practice 28
 - 1.6 Meet Python 29
 - 1.7 Our First Interaction with Python 31
 - 1.8 Keywords 31
 - 1.9 Further Reading 32
 - 1.10 Exercises 33
 - Reference 34

2	Data: The First Ingredient of a Program	35
2.1	What Is Data?	37
2.2	What Is Structured Data?	37
2.3	Basic Data Types	40
2.3.1	Integers	40
2.3.2	Floating Points	41
2.3.3	Numerical Values in Python	44
2.3.4	Characters	45
2.3.5	Boolean	48
2.4	Basic Organization of Data: Containers	50
2.4.1	Strings	50
2.4.2	Tuples	54
2.4.3	Lists	57
2.5	Accessing Data or Containers by Names: Variables	61
2.5.1	Naming	61
2.5.2	Scope and Extent	62
2.5.3	Typing	62
2.5.4	What Can We Do with Variables?	63
2.5.5	Variables in Python	64
2.6	Keywords	67
2.7	Further Reading	67
2.8	Exercises	68
3	Actions: The Second Ingredient of a Program	71
3.1	Purpose and Scope of Actions	71
3.1.1	Input-Output Operations in Python	74
3.2	Action Types	77
3.2.1	Expressions	77
3.2.2	Expressions and Operators in Python	87
3.2.3	Statements	93
3.3	Controlling Actions: Conditionals	96
3.3.1	The Turing Machine	97
3.3.2	Conditionals	99
3.3.3	Conditional Execution in Python	100
3.4	Reusable Actions: Functions	103
3.4.1	Alternative Ways to Pass Arguments to a Function	105
3.4.2	Functions in Python	108
3.5	Functional Programming Tools in Python	113
3.5.1	List Comprehension in Python	113
3.5.2	Filtering, Mapping and Reduction	113
3.6	Scope in Python	114
3.7	Keywords	115
3.8	Further Reading	116
3.9	Exercises	116

4	Managing the Size of a Problem	121
4.1	An Action Wizard: Recursion	121
4.1.1	Four Golden Rules for Brewing Recursive Definitions	125
4.1.2	Applying the Golden Rules: An Example with Factorial	127
4.1.3	Applying the Golden Rules: An Example with List Reversal	130
4.1.4	A Word of Caution About Recursion	130
4.1.5	Recursion in Python	133
4.2	Step by Step Striving Towards a Solution: Iteration	137
4.2.1	Tips for Creating Iterative Solutions	141
4.2.2	Iterative Statements/Structures in Python	142
4.3	Recursion Versus Iteration	146
4.3.1	Computational Equivalence	146
4.3.2	Resource-Wise Efficiency	146
4.3.3	From the Coder's Viewpoint	147
4.4	Keywords	148
4.5	Further Reading	148
4.6	Exercises	148
5	A Measure for 'Solution Hardness': Complexity	151
5.1	Time and Memory Complexity	151
5.1.1	Time Function, Complexity and Time Complexity	152
5.1.2	Memory Complexity	158
5.1.3	A Note on Some Discrepancies in the Use of Big-Oh	159
5.2	Keywords	160
5.3	Further Reading	160
5.4	Exercises	160
6	Organizing Data	165
6.1	Primitive and Composite Data Types	166
6.2	Abstract Data Types	166
6.2.1	Stack	166
6.2.2	Queue	167
6.2.3	Priority Queue (PQ)	168
6.2.4	Bag	170
6.2.5	Set	170
6.2.6	List	171
6.2.7	Map	171
6.2.8	Tree	173
6.3	Abstract Data Types in Python	183
6.3.1	Associative Data (Dictionaries) in Python	183
6.3.2	Stacks in Python	186
6.3.3	Queues in Python	187
6.3.4	Priority Queues in Python	188
6.3.5	Trees in Python	189
6.4	Keywords	192
6.5	Further Reading	192
6.6	Exercises	192

7	Objects: Reunion of Data and Action	195
7.1	The Idea Behind the Object-Oriented Paradigm (OOP)	196
7.2	Properties of Object-Oriented Programming	197
7.2.1	Encapsulation	199
7.2.2	Inheritance	200
7.2.3	Polymorphism	202
7.3	Object-Oriented Programming in Python	204
7.3.1	Defining Classes in Python	204
7.3.2	Inheritance in Python	209
7.3.3	Type of Objects in Python	210
7.3.4	Operator Overloading	211
7.3.5	Example with Objects in Python: Trees	212
7.3.6	Example with Objects in Python: Stacks	213
7.4	Keywords	214
7.5	Further Reading	214
7.6	Exercises	215
Index		217

Introduction to Programming Concepts with Case
Studies in Python

Ucoluk, G.; Kalkan, S.

2012, X, 222 p., Hardcover

ISBN: 978-3-7091-1342-4