

1. Record Nr.	UNINA9910830473703321
Titolo	The nature and origin of amyloid fibrils [[electronic resource] /] / [editors, Gregory R. Bock (Organizer) and Jamie A. Goode]
Pubbl/distr/stampa	Chichester ; ; New York, : Wiley, 1996
ISBN	1-282-34797-7 9786612347979 0-470-51492-2 0-470-51493-0
Descrizione fisica	1 online resource (268 p.)
Collana	Ciba Foundation symposium ; ; 199
Altri autori (Persone)	BockeGregory GoodeJamie
Disciplina	612.015782 612.8 612.8042
Soggetti	Amyloid Amyloid beta-protein Amyloidosis
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	"Symposium on The nature and origin of amyloid fibrils, held at the Palacio dos Marqueses de Pombal, Oeiras, Portugal, 23-25 October 1995"--Contents.
Nota di bibliografia	Includes bibliographical references and indexes.
Nota di contenuto	THE NATURE AND ORIGIN OF AMYLOID FIBRILS; Contents; Participants; Preface; Introduction; In memoriam; A molecular model of the amyloid fibril; Refined fibril structures: the hydrophobic core in Alzheimer's amyloid b-protein and prion as revealed by X-ray diffraction; General discussion I; Modulating conformational factors in transthyretin amyloid; Proteoglycans and amyloid fibrillogenesis; Molecular mechanisms of fibrillogenesis and the protective role of amyloid P component: two possible avenues for therapy; General discussion II; Metabolism of amyloid proteins Alzheimer's disease: genesis of amyloidApolipoprotein E and amyloidogenesis; Interaction of transthyretin with amyloid B-protein: binding and inhibition of amyloid formation; General discussion III; B-amyloid precursor protein and early-onset Alzheimer's disease; Prion

protein amyloid: separation of scrapie infectivity from PrP polymers; General discussion IV; Ageing and amyloid fibrillogenesis: lessons from apolipoprotein AI, transthyretin and islet amyloid polypeptide; General discussion V
FAP mutations destabilize transthyretin facilitating conformational changes required for amyloid formation
Index of contributors; Subject index

Sommario/riassunto

Amyloid fibrils are associated with a range of pathological disorders including Alzheimer's Disease, Down's syndrome, diabetes, cardiomyopathies, and transmissible spongiform encephalopathies. This volume is a comprehensive account of recent developments in the understanding of the process of amyloid fibrils. Contains up-to-date data on all of the clinical problems which, despite their pathological significance, are still largely unsolved.

2. **Record Nr.**

UNINA9910915680903321

Autore

Bhasin Harsh

Titolo

Python Programming Using Problem Solving

Pubbl/distr/stampa

Bloomfield : , : Mercury Learning & Information, , 2023
©2023

ISBN

1-68392-861-X
1-68392-860-1

Edizione

[1st ed.]

Descrizione fisica

1 online resource (601 pages)

Disciplina

005.133

Soggetti

Python (Computer program language)
COMPUTERS / General

Lingua di pubblicazione

Inglese

Formato

Materiale a stampa

Livello bibliografico

Monografia

Nota di contenuto

Cover -- Half-Title -- Title -- Copyright -- Dedication -- Content -- Preface -- Section I: Algorithmic Problem-Solving and Python Fundamentals -- Chapter 1: Algorithmic Problem-Solving -- 1.1 Introduction -- 1.2 Definition and Characteristics -- 1.3 Notations: Pseudocode and Flow Chart -- 1.4 Strategies for Problem-Solving:

Recursion Versus Iteration -- 1.5 Asymptotic Notation -- 1.6
Complexity -- 1.7 Illustrations -- 1.7.1 Minimum in a List -- 1.7.2
Insert a Card in a Pack of Cards (Or Insert an element in a sorted list).
There are ten cards in the pack, numbered from 1 to 10. -- 1.7.3
Guess a Number in a Given Range -- 1.7.4 Tower of Hanoi -- 1.8
Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple
Choice Questions -- Theory -- Application -- Chapter 2: Introduction
to Python -- 2.1 Introduction -- 2.2 Features of Python -- 2.2.1 Easy
-- 2.2.2 Type and Run -- 2.2.3 Syntax -- 2.2.4 Mixing -- 2.2.5
Dynamic Typing -- 2.2.6 Built-in Object Types -- 2.2.7 Numerous
Libraries and Tools -- 2.2.8 Portable -- 2.2.9 Free -- 2.3 The
Paradigms -- 2.3.1 Procedural -- 2.3.2 Object-Oriented -- 2.3.3
Functional -- 2.4 Chronology and Uses -- 2.4.1 Chronology -- 2.4.2
Uses -- 2.5 Installation of Anaconda -- 2.6 Implementation of an
Algorithm: Statement, State, Control Blocks, and Functions -- 2.6.1
Statement -- 2.6.2 State -- 2.6.3 Control Flow -- 2.7 Conclusion --
Glossary -- Points to Remember -- Resources -- Exercises -- Multiple
Choice Questions -- Theory -- Chapter 3: Fundamentals -- 3.1
Introduction -- 3.2 Basic Input Output -- 3.2.1 Print Function -- 3.2.2
Input -- 3.3 Running a Program -- 3.3.1 Using the Command Prompt
-- 3.3.2 Executing Programs Written in .py Files -- 3.3.3 Using
Anaconda Navigator -- 3.4 The Jupyter Notebook -- 3.5 Value Type
and Reference Type -- 3.6 Tokens, Keywords, and Identifiers -- 3.6.1
Python Keywords.
3.6.2 Python Identifiers -- 3.6.3 Python Escape Sequence -- 3.7
Statements -- 3.7.1 Expression Statement -- 3.7.2 Assignment
Statements -- 3.7.3 The Assert Statements -- 3.7.4 The Pass
Statements -- 3.7.5 The Control Statements -- 3.8 Comments -- 3.9
Operators -- 3.10 Types and Examples of Operators -- 3.10.1
Arithmetic Operators -- 3.10.2 String Operators -- 3.10.3 Comparison
Operators -- 3.10.4 Assignment Operators -- 3.10.5 Logical Operators
-- 3.10.6 Priority of Operators -- 3.11 Basic Data Types -- 3.11.1
Integer -- 3.11.2 Float -- 3.11.3 String -- 3.12 Conclusion --
Exercises -- Multiple Choice Questions -- Theory -- Explore -- Section
II: Procedural Programming -- Chapter 4: Conditional Statements --
4.1 Introduction -- 4.2 "If," If-Else, and If-Elif-Else Constructs -- 4.3
The If-Elif-Else Ladder -- 4.4 Logical Operators -- 4.5 The Ternary
Operator -- 4.6 The Get Construct -- 4.7 Examples -- 4.8 Summary --
Glossary -- Points to Remember -- Exercises -- Multiple Choice
Questions -- Programming Exercises -- Chapter 5: Looping -- 5.1
Introduction -- 5.2 While -- 5.3 Patterns -- 5.4 Nesting and
Applications of Loops in Lists -- 5.5 Conclusion -- Glossary -- Points
to Remember -- Exercises -- Multiple Choice Questions --
Programming Exercises -- Chapter 6: Functions -- 6.1 Introduction --
6.2 Features of a Function -- 6.2.1 Modular Programming -- 6.2.2
Reusability of Code -- 6.2.3 Manageability -- 6.2.3.1 Easy debugging
-- 6.2.3.2 Efficient -- 6.3 Basic Terminology -- 6.3.1 Name of a
Function -- 6.3.2 Arguments -- 6.3.3 Return Value -- 6.4 Definition
and Invocation -- 6.4.1 Working -- 6.5 Types of Function -- 6.5.1
Arguments: Types of Arguments -- 6.6 Implementing Search -- 6.7
Scope -- 6.8 Recursion -- 6.8.1 Rabbit Problem -- 6.8.2
Disadvantages of Using Recursion -- 6.9 Conclusion -- Glossary --
Points to Remember -- Exercises.
Multiple Choice Questions -- Programming Exercises -- Questions
Based on Recursion -- Theory -- Extra Questions -- Chapter 7: File
Handling -- 7.1 Introduction -- 7.2 The File Handling Mechanism --
7.3 The Open Function and File Access Modes -- 7.4 Python Functions
for File Handling -- 7.4.1 The Essential Ones -- 7.4.2 The OS Methods

-- 7.4.3 Miscellaneous Functions and File Attributes -- 7.5 Command Line Arguments -- 7.6 Implementation and illustrations -- 7.7 Conclusion -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Programming Exercises -- Chapter 8: Lists, tuple, and Dictionary -- 8.1 Introduction -- 8.2 Lists -- 8.2.1 Accessing Elements: Indexing and Slicing -- 8.2.2 Mutability -- 8.2.3 Operators -- 8.2.4 Traversal -- 8.2.5 Functions -- 8.3 Tuple -- 8.3.1 Accessing Elements of a Tuple -- 8.3.2 Nonmutability -- 8.3.3 Operators -- 8.3.4 Traversal -- 8.3.5 Functions -- 8.4 Associate Arrays and Dictionaries -- 8.4.1 Displaying Elements of a Dictionary -- 8.4.2 Some Important Functions of Dictionaries -- 8.4.2.1 The len function returns the number of elements in a given dictionary. -- 8.4.2.2 The max function returns the key with maximum value. If the key is a string, then the value in the lexicographic ordering would be returned. -- 8.4.2.3 The min function returns the key with minimum value. If the key is a string, then the value in the lexicographic ordering would be returned. -- 8.4.2.4 The sorted function would sort the elements of a given dictionary by their keys. If the keys are strings then lexicographic ordering would be followed. -- 8.4.2.5 The pop function takes out the element with the given key from the dictionary. -- 8.4.3 Input from the User -- 8.5 Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Programming Exercises.

Chapter 9: Iterations, Generators, and Comprehensions -- 9.1 Introduction -- 9.2 The Power of "For -- 9.3 Iterator -- 9.4 Defining an Iterable Object -- 9.5 Generators -- 9.6 Comprehensions -- 9.7 Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Programming Exercises -- Chapter 10: Strings -- 10.1 Introduction -- 10.2 Loops Revised -- 10.3 String Operators -- 10.3.1 The Concatenation Operator (+) -- 10.3.2 The Replication Operator (*) -- 10.3.3 The Membership Operator -- 10.4 In-Built Functions -- 10.4.1 len() -- 10.4.2 Capitalize() -- 10.4.3 Find() -- 10.4.4 Count -- 10.4.5 endswith() -- 10.4.6 encode -- 10.4.7 decode -- 10.4.8 Miscellaneous Functions -- 10.5 Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Section III: Object-Oriented Programming -- Chapter 11: Introduction to Object-Oriented Paradigm -- 11.1 Introduction -- 11.2 Creating New Types -- 11.3 Attributes and Functions -- 11.3.1 Attributes -- 11.3.2 Functions -- 11.4 Elements of Object-Oriented Programming -- 11.4.1 Class -- 11.4.2 Object -- 11.4.3 Encapsulation -- 11.4.4 Data Hiding -- 11.4.5 Inheritance -- 11.4.6 Polymorphism -- 11.4.7 Reusability -- 11.5 Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Explore and Design -- Chapter 12: Classes and Objects -- 12.1 Introduction to Classes -- 12.2 Defining a Class -- 12.3 Creating an Object -- 12.4 Scope of Data Members -- 12.5 Nesting -- 12.6 Constructor -- 12.7 Multiple __Init__(s) -- 12.8 Destructors -- 12.9 Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Programming Exercises -- Chapter 13: Inheritance -- 13.1 Introduction to Inheritance and Composition -- 13.1.1 Inheritance and Methods -- 13.1.2 Composition. 13.2 Inheritance: Importance and Types -- 13.2.1 Need for Inheritance -- 13.2.2 Types of Inheritance -- 13.2.2.1 Simple inheritance -- 13.2.2.2 Hierarchical inheritance -- 13.2.2.3 Multilevel inheritance -- 13.2.2.4 Multiple inheritance and hybrid inheritance -- 13.3 Methods -- 13.3.1 Bound Methods -- 13.3.2 Unbound Method -- 13.3.3 Methods are Callable Objects -- 13.3.4 The Importance and Usage of

Super -- 13.3.5 Calling the Base Class Function Using Super -- 13.4 Search in Inheritance Tree -- 13.5 Class Interface and Abstract Classes -- 13.6 Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Programming Exercises -- Chapter 14: Operator Overloading -- 14.1 Introduction -- 14.2 `__init__` Revisited -- 14.2.1 Overloading `__init__` (Sort of) -- 14.3 Methods for Overloading Binary Operators -- 14.4 Overloading Binary Operators: The Fraction Example -- 14.5 Overloading the `+=` Operator -- 14.6 Overloading the `>` and `<` Operators -- 14.7 Overloading the `__Bool__` Operator: Precedence of `__Bool__` Over `__Len__` -- 14.8 Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Programming Exercises -- Chapter 15: Exception Handling -- 15.1 Introduction -- 15.2 Importance and Mechanism -- 15.2.1 An Example of Try/Except -- 15.2.2 Manually Raising Exceptions -- 15.3 Build-in Exceptions in Python -- 15.4 The Process -- 15.4.1 Example -- 15.4.2 Exception Handling: Try/Except -- 15.4.3 Raising Exceptions -- 15.5 Crafting User Defined Exceptions -- 15.6 An Example of Exception Handling -- 15.7 Conclusion -- Glossary -- Points to Remember -- Exercises -- Multiple Choice Questions -- Theory -- Programming Exercises -- Section IV: Numpy, Pandas, and Matplotlib -- Chapter 16: Numpy-I -- 16.1 Introduction -- 16.2 Fundamentals. 16.2.1 Similarity and Differences Between a List and a NumPy Array.

Sommario/riassunto

Python is a robust, procedural, object-oriented, and functional language. The features of the language make it valuable for web development, game development, business, and scientific programming. This book deals with problem-solving and programming in Python. It concentrates on the development of efficient algorithms, the syntax of the language, and the ability to design programs in order to solve problems. In addition to standard Python topics, the book has extensive coverage of NumPy, data visualization, and Matplotlib. Numerous types of exercises, including theoretical, programming, and multiple-choice, reinforce the concepts covered in each chapter. FEATURES: Concentrates on the development of efficient algorithms, the syntax of the language, and the ability to design programs in order to solve problems. Features both standard Python topics and also extensive coverage of NumPy, data visualization, and Matplotlib problem-solving techniques
