TT TEXTBOOK TESTBANKS

Solutions Manual for Python Fundamentals 1st Cengage

SOLUTIONS MANUAL SAMPLE PREVIEW

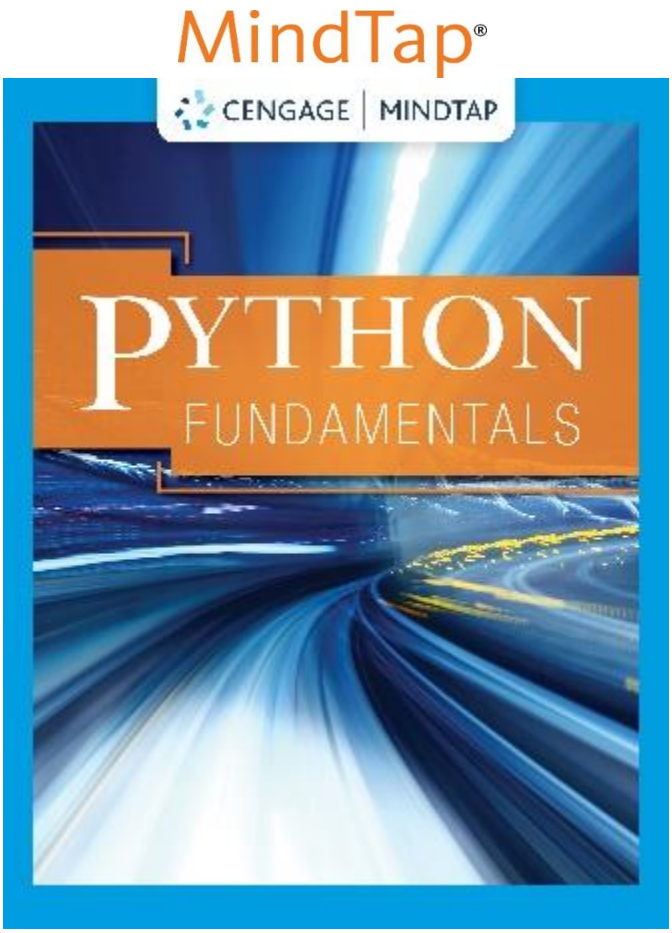
PUBLISHER
Cengage

COPYRIGHT
2021

RESOURCE TYPE
Solutions Manual

ISBN
9780357421796

Python Fundamentals
ISBN MindTap:



Welcome to *Python Fundamentals*. This Instructor's Manual will help you navigate the unique activities that are included in the MindTap, which will better enable you to include the exercises in your curriculum. While the content included in this MindTap is specific to the discipline and course, the functionality will act the same as you move from product to product.

For additional resources on our MindTap platform, please click HERE. At this site, you will find User Guides, Self-Training Videos, Training Webinars, and Podcasts. We also include Resources that are specific to your campus's LMS, should additional information be needed. Student versions of the same resources are located HERE. This link can be shared with your students directly, should they have any questions about the product.

At a Glance

Instructor's Manual Table of Contents

Course Learning Design

In creating the digital learning path, we aimed to provide your students with a coherently structured experience that:

- supports and aligns the learning objectives with the course content, instructional strategies, and assessments;
- addresses individual learner differences and preferences;
- welcomes learners of all abilities and backgrounds; and
- enhances learner motivation by providing them with relevant, applicable learning experiences consistent with their own learning and professional goals.

We're excited to present you with the digital course experience and want to draw your attention to some of the design decisions we made as part of ensuring your confidence in our ability to create an effective, quality learning experience.

Course Learning Design	
Course Description	As you work with the language, you'll learn about control statements, delve into controlling program flow, and gradually work on more structured programs via functions. *MindTap for Python Fundamentals* teaches problem-solving skills for building efficient applications. As you settle into the Python ecosystem, you'll learn about data structures and study ways to correctly store and represent information. By working through specific examples, you'll learn how Python implements object-oriented programming (OOP) concepts of abstraction, encapsulation of data, inheritance, and polymorphism. Coverage also includes an overview of how imports, modules, and packages work in Python, how you can handle errors to prevent apps from crashing, as well as file manipulation.
Course Approach (9 modules in course)	This course teaches students how to write systematic code in Python and improve application efficiency with hands-on practice, step-by-step instruction, and provides immediate feedback and troubleshooting support on their code. Students will develop skills that are in-demand by employers by completing authentic, real-world coding projects that can be added to their GitHub portfolios.
Module Approach	Each module is broken into 2–6 lessons—within each lesson are activities that align to meet specific learning objectives that are concrete and actionable. Within each lesson, the student will read some narrative and follow up with hands-on learning. There are four types of online labs in this course: 1. **Practice Exercises** (Ungraded) provide an opportunity to practice a new concept in a short coding activity. Students are provided with guided instructional materials alongside a live computing environment. There will typically be 1–3 practice labs in each lesson and there are on average around 5 lessons per module (around 5 practice/module). 2. **Lab Activities** (Auto-Graded) are coding activities that are completed by a student and contain auto-grading that feeds directly to the gradebook. Learners demonstrate an understanding of numerous concepts by completing tasks. Tasks are verified using unit tests, I/O tests, image and webpage comparison, debugging tests, and many other checks. There will be a lab assessment for every lesson and there are on average around 5 lessons per module (around 5 labs per module). 3. **Module Lab Assessments** (Auto- and Manual-Graded) encompass all the learning objectives in the module. Students are asked to complete a larger, authentic assignment with many tasks. Some tasks will be verified using unit tests, I/O tests, image and webpage comparison, debugging tests but other tasks will be unique to each student's project and will require manual grading. The goal of these assignments is to prove that students have mastered the learning objectives in the module and in doing so have also created a program for their GitHub portfolio (1 Module Lab Assessment per module). 4. **Capstone Lab Assessment (**Auto- and Manual-Graded) is a final project that is the summative assessment. The goal of this assignment is to prove that students have mastered the course objectives and in doing so have also created a program for their GitHub portfolio (1 Capstone Lab Assessment per course).

Learning Path Activities	How many in course	What is it?	Why it matters?	Seat time
Welcome to Your Course	1	This is a brief overview of the course objectives that will be covered in the modules of this MindTap.	Students will gain a clear understanding of the course objectives and will explore how this course offers the opportunity to not only read but watch videos, engage in critical-thinking simulations and hands-on trainings, teach them how to use the technology, and take quizzes to practice and check their understanding.	5 minutes
Getting Started Resources	1	This section includes videos that provide an overview of the MindTap platform and the Coding IDE. There are 3 lab Pre-Requisites, 2 of which count toward the grade.	Students will learn how to use MindTap to its fullest potential, which will help them excel in the course. They'll also be introduced to the IDE's functionality in 6 brief videos. They'll then complete 3 Lab Pre-Requisites, 1 is practice and 2 count toward their grade.	30 minutes
Pre- and Post-Course Assessments	27 questions each assessment	Brief survey to-assess students' knowledge of the subject matter before and after completing the course.	**For students:** It creates awareness around what they will learn (pre) and how much they have learned (post). **For instructors:** It establishes a baseline of what students already know (pre) and demonstrates how much they learned (post). **For administrators:** Coupling the pre- and post-course assessment provides data on how much the students learned and the overall impact of the course.	40 minutes
Module Content (9 modules total)				
Readings for each module lesson; 2–6 lessons per module	~7 Short readings per module (69 total in course)	Readings reinforce learning objectives.	Students will read succinct, focused excerpts vs long chapters (then move into an interactive activity).	55 minutes
Practice Exercises	~5 per module (48 total in course)	Short coding exercises in an IDE (non-graded)	Students complete step-by-step coding exercises that offer a practical, hands-on approach to acquiring and retaining new concepts and skills.	2-5 minutes
Lab Activity (Graded)	~5 per module (41 total in course)	Scenario-based coding labs in an IDE (auto-graded)	These scenario-based activities bring together skills learned throughout the topics and lessons to solve real-world problems.	30 minutes
Reflection	~6 per module (51 total in course)	Essay question	The reflection prompt challenges students to develop higher-level thinking and promotes problem-solving. This is also an opportunity for you to confirm that tricky topics are understood.	15 minutes
Module Quizzes	~1 per module (9 total in course)	Includes 10 multiple-choice questions at the end of each module.	The student can integrate material across the entire lesson and check their understanding before moving on to the next lesson.	10 minutes
Module Lab Assessment (Auto & Manual Grading)	1 per module (9 total in course)	A larger coding project in our IDE that assesses whether students have mastered the Learning Objectives in the module.	A larger lab with an authentic development project with many tasks. Upon completion, students will have 9 large coding projects for their GitHub portfolios.	1–2 hours
Capstone Lab Assessment	1 per course	Final coding project in our IDE that assesses whether	A larger lab with an authentic development project with many tasks. Upon completion, students will have	2–5 hours

		students have mastered the Course Objectives.	1 additional coding project to add to their GitHub portfolio.	
Instructor Test Bank	1 per module (9 total in course)	An exam of 451 objective-based questions based on each module available in the CNOW app.	The Test Bank evaluates the student on their mastery of that module.	30 minutes

Topic/Chapter	Assignments
Module 1 Introducing Python	Lessons 1.1 – 1.4 Reading Practice Exercises Lab Activities Reflection Module Quiz
Module 2 Data Types	Lessons 2.1 – 2.4 Reading Practice Exercises Lab Activities Reflection Module Quiz
Module 3 Control Statements	Lessons 3.1 – 3.9 Reading Practice Exercises Lab Activities Reflection Module Quiz
Module 4 Functions	Lessons 4.1 – 4.4 Reading Practice Exercises Lab Activities Reflection Module Quiz
Module 5 Lists and Tuples	Lessons 5.1 – 5.6 Reading Practice Exercises Lab Activities Reflection Module Quiz
Module 6 Dictionaries and Sets	Lessons 6.1 – 6.6 Reading Practice Exercises Lab Activities Reflection Module Quiz
Module 7 Object-Oriented Programming	Lessons 7.1 – 7.7 Reading Practice Exercises Lab Activities Reflection Module Quiz
Module 8 Modules, Packages, and File Operations	Lessons 8.1 – 8.7 Reading Practice Exercises Lab Activities Reflection Module Quiz
Module 9 Error Handling	Lessons 9.1 – 9/4 Reading Practice Exercises Lab Activities Reflection Module Quiz
Capstone Lab Assessment:	

Unit Testing Rest APIs	

Lab Details

There are 48 Practice Exercises, 41 Lab Activities, 9 Module Lab Assessments, and 1 Capstone Lab Assessment across 9 modules.

Lab Types

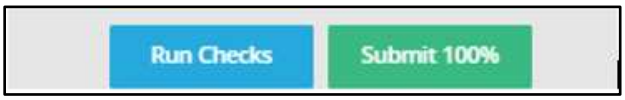
Practice Exercises:

- Practice Exercises are coding lab assignments within the IDE that allow you to practice writing and running code.
- Practice Exercises are not graded and are not captured in the Progress App. These are designated in the learning path:

Lab Activities:

- Lab Activities are coding lab assignments within the IDE that run tests against your code to ensure that the objectives in the activity have been satisfied.
- Lab Activities are automatically graded unless otherwise noted in the learning path as "PRACTICE". All graded labs are designated in the learning path as "COUNTS TOWARDS GRADE".

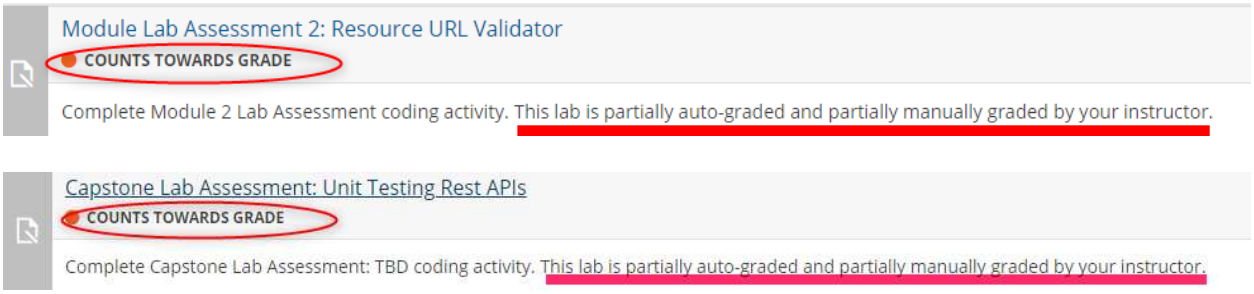
- You will work through the Lab Activities and "Run Checks" as you work through the problems. Once you have completed the assignment, you can "Submit", which will send your lab to your instructor.



- Note that instructors have the capability to review code submissions and alter grades as they see fit. Grade submissions are not final.

Module Lab Assessments and Capstone Lab Assessment:

- The Lab Assessments are coding lab assignments within the IDE that provide you with an authentic scenario to test your coding skills.
- There is one Module Lab Assessment per module, and one Capstone Lab Assessment for the entire course.
- Lab Assessments are partially automatically graded and partially manually graded by your instructor. These are designated in the learning path with a "This lab is partially auto-graded and partially manually graded by your instructor" description. All graded labs are designated in the learning path as "COUNTS TOWARDS GRADE".



- Note that instructors have the capability to review code submissions and alter grades as they see fit. Grade submissions are not final.

List of Coding Labs

Coding IDE Lab Prerequisite for Practice Exercises	Practice
Coding IDE Lab Prerequisite for Lab Activities	Auto-Grade
Coding IDE Lab Prerequisite for Module and Capstone Lab Assessments	Auto / Manual Grade
Module 1	
Practice Exercise 1.1A: Checking our Python Installation	Practice
Practice Exercise 1.1B: Working with the Python Interpreter	Practice
Lab Activity 1.1: Working with the Python Shell	Auto-Grade

Practice Exercise 1.2: Creating a Script	Practice
Lab Activity 1.2: Running Simple Python Scripts	Auto-Grade
Practice Exercise 1.3A: Checking the Type of a Value	Practice
Practice Exercise 1.3B: Using Variables	Practice
Lab Activity 1.3A: Using Variables and Assign Statements	Auto-Grade
Practice Exercise 1.3C: Python Keywords	Practice
Lab Activity 1.3B: Variable Assignment and Variable Naming Conventions	Auto-Grade
Practice Exercise 1.4A: Fetching and Using User Input	Practice
Practice Exercise 1.4B: The Importance of Proper Indentation	Practice
Lab Activity 1.4A: Fixing Indentations in a Code Block	Auto-Grade
Lab Activity 1.4B: Implementing User Input and Comments in a Script	Auto-Grade
Module Lab Assessment 1: Creating a Unit Converter	Auto / Manual Grade
Module 2	
Practice Exercise 2.1: Converting Between Different Types of Number Systems	Practice
Lab Activity 2.1A: Order of Operations	Auto-Grade
Lab Activity 2.1B: Using Different Arithmetic Operators	Auto-Grade
Lab Activity 2.2A: String Slicing	Practice
Lab Activity 2.2B: Working with Strings	Auto-Grade
Practice Exercise 2.2: Using Escape Sequences	Practice
Lab Activity 2.2C: Manipulating Strings	Auto-Grade
Practice Exercise 2.3: List References	Practice
Lab Activity 2.3: Working with Lists	Auto-Grade
Lab Activity 2.4: Using Boolean Operators	Auto-Grade
Module Lab Assessment 2: Resource URL Validator	Auto / Manual Grade
Module 3	
Practice Exercise 3.2: Using the if Statement	Practice
Lab Activity 3.2: Working with the if Statement	Auto-Grade
Practice Exercise 3.3A: Using the while Statement	Practice
Practice Exercise 3.3B: Using while to Keep a Program Running	Practice
Lab Activity 3.3: Working with the while Statement	Auto-Grade
Practice Exercise 3.6: Using the for Loop	Practice
Lab Activity 3.7: The for Loop and the range Function	Auto-Grade
Practice Exercise 3.8: Using Nested Loops	Practice
Lab Activity 3.8: Nested Loops	Auto-Grade
Lab Activity 3.9: Breaking Out of Loops	Auto-Grade
Module Lab Assessment 3: Abby's Ice Cream Shop	Auto / Manual Grade
Module 4	
Practice Exercise 4.2: Defining Global and Local Variables	Practice
Lab Activity 4.3: Function Arguments	Auto-Grade

Practice Exercise 4.4: Creating a Lambda Function	Practice
Lab Activity 4.4: Using Lambda Functions	Auto-Grade
Module Lab Assessment 4: AppInvest Return on Investment Function	Auto / Manual Grade
Module 5	
Lab Activity 5.2: Using the List Methods	Auto-Grade
Practice Exercise 5.4: Creating a Tuple	Practice
Practice Exercise 5.5A: Accessing Tuple Elements Using Indexing	Practice
Practice Exercise 5.5B: Accessing Tuple Elements Using Slicing	Practice
Lab Activity 5.6: Using Tuple Methods	Auto-Grade
Module Lab Assessment 5: Creating a Blackjack Simulator	Auto / Manual Grade
Module 6	
Lab Activity 6.1A: Creating a Dictionary	Auto-Grade
Practice Exercise 6.1: Adding, Reading, and Iterating through a Dictionary	Practice
Lab Activity 6.1B: Arranging and Presenting Data Using Dictionaries	Auto-Grade
Lab Activity 6.1C: Combining Dictionaries	Auto-Grade
Practice Exercise 6.2: Updating, Editing, and Copying from a Dictionary	Practice
Lab Activity 6.4: Building a Set	Auto-Grade
Practice Exercise 6.4: Adding, Reading, Editing, and Building a Set	Practice
Lab Activity 6.5: Creating Unions of Elements in a Collection	Auto-Grade
Practice Exercise 6.5: Unions, Differences, and Intersection of Sets	Practice
Practice Exercise 6.6: Frozen Sets	Practice
Module Lab Assessment 6: Space Explorer (Dictionaries and Sets)	Auto / Manual Grade
Module 7	
Practice Exercise 7.2: Adding Attributes to a Class	Practice
Lab Activity 7.2: Defining a Class and Objects	Auto-Grade
Lab Activity 7.3: Defining Methods in a Class	Auto-Grade
Practice Exercise 7.4A: Declaring a Class with Instance Attributes	Practice
Practice Exercise 7.4B: Implementing a Counter for Instances of a Class	Practice
Lab Activity 7.4: Creating Class Attributes	Auto-Grade
Practice Exercise 7.5A: Testing our Factory Method	Practice
Practice Exercise 7.5B: Accessing Class Attributes from within Class Methods	Practice
Lab Activity 7.5: Creating Class Methods and Using Information Hiding	Auto-Grade
Practice Exercise 7.6: Implementing Class Inheritance	Practice
Lab Activity 7.6: Overriding Methods	Auto-Grade
Practice Exercise 7.7: Implementing Multiple Inheritance	Practice

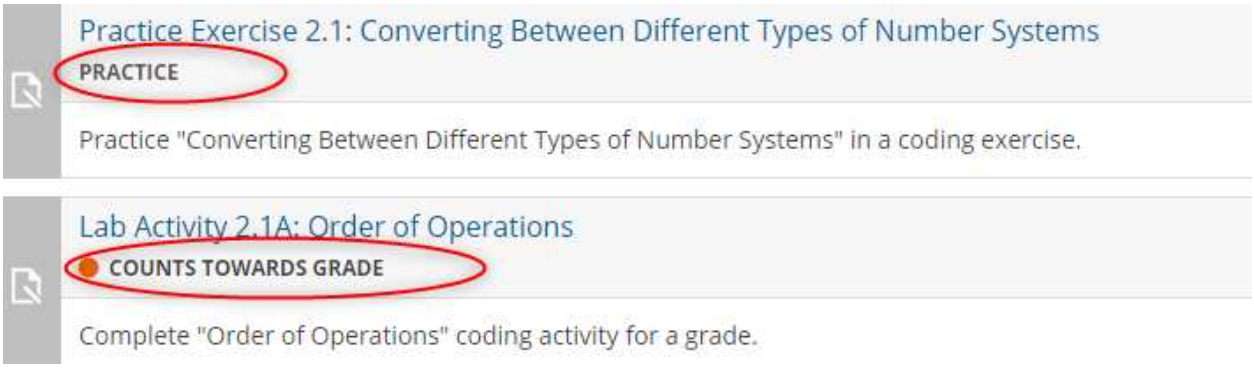
Lab Activity 7.7: Practicing Multiple Inheritance	Auto-Grade
Module Lab Assessment 7: Petri Dish Simulation (Object-Oriented Programming)	Auto / Manual Grade
Module 8	
Practice Exercise 8.1: Creating and Importing a User-Defined Module	Practice
Practice Exercise 8.2A: Importing Modules	Practice
Practice Exercise 8.2B: Importing Functions from User-Defined Modules	Practice
Practice Exercise 8.3: Inspecting Modules and Packages	Practice
Lab Activity 8.3A: Inspecting Modules	Auto-Grade
Lab Activity 8.3B: Listing the Resources Defined in a Package or Module	Auto-Grade
Lab Activity 8.3C: Using Resources in a Module	Auto-Grade
Practice Exercise 8.6A: Creating and Writing to a Text File	Practice
Practice Exercise 8.6B: Read Using with Keyword	Practice
Practice Exercise 8.6C: File Operations	Practice
Lab Activity 8.6: Performing File Operations	Auto-Grade
Practice Exercise 8.7A: Reading a CSV file	Practice
Practice Exercise 8.7B: Write a dict to CSV	Practice
Lab Activity 8.7: Working with Files	Auto-Grade
Practice Exercise 8.7C: Working with JSON	Practice
Module Lab Assessment 8: Mailing List Validation File Processing	Auto / Manual Grade
Module 9	
Practice Exercise 9.1A: Raise an Exception	Practice
Practice Exercise 9.1B: Raise an Exception with the raise Keyword	Practice
Lab Activity 9.2: Identifying Error Scenarios	Auto-Grade
Practice Exercise 9.3A: Implement a try...except Block	Practice
Practice Exercise 9.3B: Implementing the try…except…else Block	Practice
Lab Activity 9.3: Handling Errors	Auto-Grade
Practice Exercise 9.4: Catch an Error and Raise an Exception	Practice
Lab Activity 9.4: Creating Your Own Custom Exception Class	Auto-Grade
Module Lab Assessment 9: Error Handling	Auto / Manual Grade
Capstone Lab Assessment: Unit Testing Rest APIs	Auto / Manual Grade

A Note to Instructors:

COUNTS TOWARD GRADE/PRACTICE: Whether a lab COUNTS TOWARD GRADE or is PRACTICE, as indicated in the Learning Path, is preset and cannot be changed. Changing the Gradeable field within MindTap will not change the gradeability of the actual labs. We

recommend not changing these default settings to avoid discrepancies between the MindTap plank description and the actual lab.

- COUNTS TOWARD GRADE = lab is automatically or manually graded and the score is captured in the Progress App
- PRACTICE = lab is not graded and the score is not captured in the Progress App



Module 1

Introducing Python

Module Objectives

- Use the Python interactive shell to write simple programs
- Write and run simple Python scripts
- Write and run dynamic scripts that take arguments from the command line
- Use variables and describe the different types of values that variables can be
- assigned
- Get user input from the keyboard for your Python programs
- Explain the importance of comments and write them in Python
  Explain the importance of whitespace and indentation in Python

Solutions to Reflection

Lesson 1.1 Reflection

1. Which other operating system shells are you familiar with?

   ANS: Bourne-again shell (Bash), Z shell, and Korn shell for Unix-based systems such as Linux and the Windows and Windows PowerShell.

2. What are the benefits of the Python interactive shell?

   ANS: It allows you to quickly execute Python commands without having to compile anything. It also allows you to do quick tests.

Lesson 1.2 Reflection

1. What are the advantages of two different methods of running Python programs?

   ANS:
   Pros of the interactive shell:
   1) Running Python programs via the interactive shell has the benefit of being able to run quick tests/experiments as opposed to running a saved module, which has the overheads of creating a file and having to change and save it if you want to run those changes.
   2) You can run quick calculations using the Python interactive shell.

   Pros of running a saved script:
   1) You can reuse code by either running the same script or importing it into your module.

2) You can automate different tasks and have them run on your system on a schedule.

Lesson 1.3 Reflection

1. Why do we get a syntax error when we try to assign reserved words?

   ANS: The Python interpreter tries to parse the keyword's syntactical meaning, which doesn't fit into the context of a variable assignment. For example, when you try assigning the keyword if, Python looks for a condition next but instead finds an equals sign, which is invalid.

2. Mark the following variable names as valid or invalid and state why.

1) TestTOKEN
2) new-list
3) __
4) TOTAL
5) if
6) array2
7) 2_nd
8) void
9) five%
10) n123456789
11) LastLetter
12) maximum width
13) $HOME

ANS: Valid variable names: 1, 3, 4, 6, 8, 10, 11. Invalid variable names: 2, 5, 7, 9, 12, 13.

Lesson 1.4 Reflection

1. How does Python indentation help while writing scripts and code?

   ANS: The indentation in Python can increase the clarity of the code by making it obvious to a human reader which blocks of code belong together.