UnCodemy

Python Programming Important Questions PDF

The most important Python Programming questions are Python features and data types, mutable versus immutable objects, operators and control flow, loops with break and continue, functions with default, keyword and variable arguments, recursion, lambda, map and filter, strings and slicing, lists, tuples, sets and dictionaries with comprehensions, file handling, exception handling, classes, inheritance, modules and packages, and libraries such as NumPy and Pandas.

Key takeaways

- Most long answers ask for a short program, so practise writing correct Python by hand.
- List versus tuple, mutable versus immutable and function arguments come up very often.
- File handling and exception handling are reliable long-answer topics.
- These are practice questions, not past papers; check your own syllabus and official previous papers.

About these important questions

This page lists the most important Python Programming questions for B.Tech CSE and IT exams, arranged unit by unit. The units follow the standard syllabus that most Indian technical universities share, so the order may differ slightly from yours.

Please note: these are practice questions based on common syllabus topics. They are not verified past papers, and we do not claim that any question will appear in your exam. Check your own university syllabus first, then solve the official previous year papers. AKTU students can download them from the AKTU old question paper archive.

Unit 1: Introduction, Data Types and Operators

Covers Python basics, the interpreter, variables, data types, operators and input/output.

Short questions (2 marks)

- List four features of Python.
- Differentiate between mutable and immutable data types with examples.
- What is the difference between == and is?
- What is dynamic typing?

Long questions

1. Explain the built-in data types in Python with examples.
2. Explain the operators in Python, including membership and identity operators, and operator precedence.
3. Explain how Python code runs: the interpreter, bytecode and the Python virtual machine.
4. Write a program to swap two numbers without a third variable, and to check whether a number is an Armstrong number.
5. Explain type conversion with int(), float() and str().

Unit 2: Control Flow and Loops

Covers conditional statements, loops and loop control.

Short questions (2 marks)

- Differentiate between break, continue and pass.
- What is the use of the else clause in a loop?
- What does range(2, 10, 3) produce?

Long questions

1. Explain if, if-else and if-elif-else with programs.
2. Compare for and while loops with examples.
3. Write programs to print the Fibonacci series, check for a prime number, and find the factorial of a number.
4. Write a program to print a pyramid pattern of stars or numbers.
5. Write a program to find the sum of digits and reverse a number.

Unit 3: Functions, Strings and Recursion

Covers user-defined functions, arguments, scope, lambda functions, recursion and string handling.

Short questions (2 marks)

- Differentiate between local and global variables.
- What are `*args` and `**kwargs`?
- What is a lambda function?

Long questions

1. Explain the types of function arguments: positional, keyword, default and variable-length, with programs.
2. Explain recursion with a program for factorial or Tower of Hanoi.
3. Explain map(), filter() and reduce() with examples.
4. Explain string slicing and ten common string methods with examples.
5. Write a program to check whether a string is a palindrome and count its vowels.

Unit 4: Lists, Tuples, Sets and Dictionaries

Covers Python's built-in collections and comprehensions. This unit is asked heavily.

Short questions (2 marks)

- Differentiate between a list and a tuple.
- What is a dictionary? How do you access its keys and values?
- What is a set? How is it different from a list?

Long questions

1. Explain list methods (append, extend, insert, remove, pop, sort) with examples.
2. Explain list comprehension and dictionary comprehension with examples.
3. Explain dictionary methods and write a program to count the frequency of words in a sentence.
4. Explain set operations: union, intersection, difference and symmetric difference.
5. Write programs for linear search, binary search and bubble sort on a list.

Unit 5: Files, Exceptions, OOP and Modules

Covers file handling, exception handling, classes and objects, inheritance, modules, packages and common libraries.

Short questions (2 marks)

- List the file opening modes in Python.
- What is the purpose of `__init__`?
- Differentiate between a module and a package.

Long questions

1. Write a program to read a text file and count its lines, words and characters.
2. Explain exception handling with try, except, else and finally, and raise a custom exception.
3. Explain classes, objects, constructors and the self parameter with a program.
4. Explain inheritance and method overriding in Python, including multiple inheritance.
5. Explain how to create and import a module, and the use of `if __name__ == "__main__"`.
6. Explain NumPy arrays and basic Pandas DataFrame operations with examples.

Sample answer outline: Types of function arguments

A typical 7-mark question is "Explain the different types of arguments in Python functions with examples." Structure the answer like this:

1. **Define a function and an argument.** A function is a named block of code defined with def. Arguments are the values passed when calling it.
2. **Positional arguments.** Values match parameters by position: `add(2, 3)`.
3. **Keyword arguments.** Values match by name, so order does not matter: `add(b=3, a=2)`.
4. **Default arguments.** A parameter has a default value: `def greet(name, msg="Hello")`. Default parameters must come after non-default ones.

5. **Variable-length arguments.** `*args` collects extra positional values into a tuple, and `**kwargs` collects extra keyword values into a dictionary.
6. **Give one combined program.** For example, `def show(a, b=10, *args, **kwargs): print(a, b, args, kwargs)`, then call it with `show(1, 2, 3, 4, x=5)` and write the output: `1 2 (3, 4) {'x': 5}`.

Writing the output of your example proves you understand how the values are matched.

Marks weightage and how to write answers

A common pattern, used by AKTU and many other universities, has a compulsory short-answer section that covers every unit, followed by long-answer questions with a choice inside each unit. Marks per question change between universities and years, so confirm the scheme on a recent official paper.

- **Short answers (about 2 marks):** a crisp definition, one example or diagram, and nothing more. Three to five lines is enough.
- **Long answers (7 to 10 marks):** start with a definition, add a labelled diagram or table, explain the steps, and finish with an example, advantages or a comparison.
- Indent Python code clearly in the answer sheet. Wrong indentation changes the meaning of the program.
- Write the expected output below every program. It shows the examiner that the logic works.

A 7-day revision plan for Python Programming

1. **Days 1 to 5:** one unit a day. Read your notes, then write the long answers for that unit from memory and check them against your notes.
2. **Day 6:** write from memory the programs for prime check, palindrome, word frequency, file line count and a class with inheritance.
3. **Day 7:** solve one full official previous year paper in three hours, then check which units cost you time.

Keep a single sheet of formulas, definitions and diagrams for the last hour before the exam.

Frequently asked questions

Which Python Programming topics are most important?

Data types, mutable versus immutable, loops, function arguments, strings, lists, tuples, dictionaries, file handling, exception handling and classes are asked most often.

How should I write Python programs in the exam?

Keep programs short, use clear variable names, indent consistently with four spaces and write the expected output. Add a one-line comment for the key step.

Is Python useful for a data science career?

Yes. Python is the main language for data analysis, machine learning and automation, so the basics you learn for this exam are directly useful.

Are these Python Programming questions from previous year papers?

No. They are practice questions built from topics that appear in most B.Tech Python Programming syllabi. Use them with your own university's official previous papers, for example the AKTU archive at https://aktu.ac.in/old-question-paper.html.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- AKTU: old question papers (official archive)
- Python documentation: The Python Tutorial
- NPTEL: online courses from IITs (search for Python)

About this PDF

These are practice questions for revision, prepared by the Uncodemy trainer team from common B.Tech syllabus topics. They are not past papers, a certificate or a promise of marks. Spot an error? Write to info@uncodemy.com and the team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550