Program	B.Pharmacy
Semester	First
Course Title /Subject Name	Basics of Python Programming for Pharmaceutical Sciences (Theory)
Course Code	BP101T
Module	Introduction to Python programming
Course Coordinator	Ms. Ramanjit Kaur
Mobile no	8968729307

Course Objectives:

The objectives of this course are to:

1. Introduce the fundamentals of python programming for pharmaceutical sciences.
2. Develop basic programming skills using control structures, functions ,and data structures.
3. Provide knowledge of data handling techniques for structured dataset management.
4. Familiarize students with data analysis tools such as NumPy and Pandas for healthcare datasets.
5. Enable students to visualize and interpret pharmaceutical data.

Course Outcomes (CO):

CO No.	Upon successful completion of this course, the students will be able to:
BP101.1	Explain the fundamentals of Python programming, including variables, data types, operators, and libraries.
BP101. 2	Analyze program logic using control structures and functions.
BP101. 3	Organize, manipulate, and retrieve data using data structures and file handling techniques.
BP101. 4	Analyze pharmaceutical datasets using Python libraries.
BP101 .5	Visualize and interpret pharmaceutical data using graphical tools.

LO	Learning Outcomes	Course Outcome Code
1	Install Python and use IDEs such as Jupyter Notebook, PyCharm, and Visual Studio Code for Python programming.	BP101.1
2	Explain the advantages of IDEs over text editors, including syntax highlighting, debugging, auto-completion, and project management features.	BP101.1
3	Create and use Python variables, data types (integers, floats, strings, booleans), type casting, arithmetic operators, comparison operators, and logical operators in programs.	BP101.1
4	Perform input and output operations and apply basic string operations and manipulation techniques such as concatenation, indexing, slicing, and string methods.	BP101.1
5	Differentiate between standard libraries and third-party libraries, and install or uninstall Python libraries using pip commands.	BP101.1

Introduction to Python programming

Python programming

Python is a general-purpose programming language in a similar to other programming languages that you might have heard of such as C++, JavaScript or Microsoft's C# and Oracle's Java. Python is a high-level, general-purpose and easy-to-learn programming language widely used in web development, data science, automation, artificial intelligence, and more. It was created by Guido van Rossum and first released in 1991. Python is a popular programming language known for being easy to read and write because its code looks like everyday English.

Python is an open source and cross-platform programming language, that has become increasingly popular over the last ten years. Python is a multi-purpose programming languages (due to its many extensions), examples are scientific computing and calculations, simulations, web development etc.

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed. Depending on the Editor you are using, this is either done automatically, or you need to do it manually.

As a language it has gained in interest over recent years, particularly within the commercial world, with many people wanting to learn the language. This increased interest in Python is driven by several different factors:

1. Python is highly flexible and simple, which makes it very easy for beginners to learn.
2. It is widely used in the Data Science community and serves as a more standardized programming language compared to some alternatives like R.
3. It is well-suited for scripting in the DevOps field, offering a higher level of abstraction than many traditional programming languages.
4. It can run on almost all operating systems, especially the major ones such as Windows, macOS, and Linux.
5. It provides a large collection of libraries (modules) that help extend and enhance its basic functionality.

Core Characteristics

- **Easy to Read**: Python uses a simple syntax that closely resembles English, making it beginner-friendly.

- **Interpreted**: It executes code line-by-line, which allows for fast debugging and testing.
- **Versatile**: It is a "general-purpose" language, meaning it can be used for a vast range of tasks rather than being specialized for just one.
- **Multi-Platform Support**: Python programs can run on operating systems such as Windows, macOS, and Linux with little to no modification.
- **Vast Libraries**: It comes with a huge “batteries-included” standard library and countless third-party packages offering ready-made tools for nearly every task.
- **User-friendly**: Because it handles complex computer tasks automatically (like memory management), beginners can focus on learning logic rather than technical details.
- **Straight forward syntax:** The formation of python syntax is simple and straight forward which also makes it popular.
- **Python is object-oriented:** Structure supports such concepts as polymorphism, operation overloading and multiple inheritance

.

Python Versions

- Currently there are two main versions of Python called Python 2 and Python 3.
- Python 2 was launched in October 2000 and has been, and still is, very widely used.
- Python 3 was launched in December 2008 and is a major revision to the language that is not backward compatible.

The issue between the two versions can be highlighted by the simple print facility:
• In Python 2 this is written as print 'Hello World'
• In Python 3 this is written as print ('Hello World')

It may not look like much of a difference but the inclusion of the '()' marks a major change and means that any code written for one version of Python will probably not run on the other version. There are tools available, such as the 2–3 utility, that will (partially) automate translation from Python 2 to Python 3 but in general you are still left with significant work to do.

Python Programming

There are several different programming paradigms that a programming language may allow developers to code in, these are:

- **Procedural Programming** in which a program is represented as a sequence of instructions that tell the computer what it should do explicitly. Procedures and/ or functions are used to provide structure to the program; with control structures such as if statements and loop constructs to manage which steps are executed and how many times. Languages typifying this approach include C and Pascal.
- **Declarative Programming** languages, such as Prolog, that allow developers to describe how a problem should be solved, with the language/environment determining how the solution **should be implemented. SQL (a database query language) is one of the most common** declarative languages that you are likely to encounter.
- **Object Oriented Programming** approaches that represent a system in terms of the objects that form that system. Each object can hold its own data (also known as state) as well as define behavior that defines what the object can do. A computer program is formed from a set of these objects co-operating together. Languages such as Java and C# typify the object oriented approach.
- **Functional Programming languages** decompose a problem into a set of functions. Each function is independent of any external state, operating only on the inputs they received to generate their outputs. The programming language Haskell is an example of a functional programming language.

Interpreted vs. Compiled

What are the differences between Interpreted programming languages and Compiled programming languages? What kind should you choose, and why should you bother? Programming languages generally fall into one of two categories: Compiled or Interpreted. With a compiled language, code you enter is reduced to a set of machine-specific instructions before being saved as an executable file.
Both approaches have their advantages and disadvantages.

With interpreted languages, the code is saved in the same format that you entered. Compiled programs generally run faster than interpreted ones because interpreted programs must be reduced to machine instructions at run-time. It is usually easier to develop applications in an interpreted environment because you don't have to recompile your application each time you want to test a small section.

Python is an interpreted programming language, while e.g., C/C++ are translated by running the source code through a compiler, i.e., C/C++ are compiled languages.

Interpreted languages, in contrast, must be parsed, interpreted, and executed each time the program is run.

Another example of an interpreted programming language is PHP, which is mainly used to create dynamic web pages and web applications.

Compiled languages are all translated by running the source code through a compiler. This results in very efficient code that can be executed any number of times. The overhead for the translation is incurred just once, when the source is compiled; thereafter, it need only be loaded and executed.

During the design of an application, you might need to decide whether to use a compiled language or an interpreted language for the application source code.

Interpreted languages, in contrast, must be parsed, interpreted, and executed each time the program is run .

Thus, an interpreted language is generally more suited for doing ”ad hoc” calculations or simulations, while compiled languages are better for permanent applications where speed is in focus.

Installing Python and an Integrated Development Environment (IDE) [Jupyter Notebook, PyCharm, VS Code]

To get started with Python development, you first need to install the Python interpreter and then choose a development environment (IDE or code editor) to write your scripts efficiently.

1. Install Python

Here are the most popular ways to install the Python language and an IDE on Windows.

Software	Best For
Jupyter Notebook	Learning, practice, data science
PyCharm	Full Python coding projects
Visual Studio Code	General coding + Python

Step 1 — Install Python

Download Python

Go to the official website:

Python official website : http://www.python.org

Download the latest version for Windows.

Important During Installation

When installing:

✅Tick this option:

Add Python to PATH

Then click:

Install Now

Check Python Installed Correctly

Open Command Prompt and type:

```
python–version
```

You should see something like:

```
Python 3.13.3
```

Option 1 — Install Jupyter Notebook (Best for Learning)

Install Jupyter

Open Command Prompt:

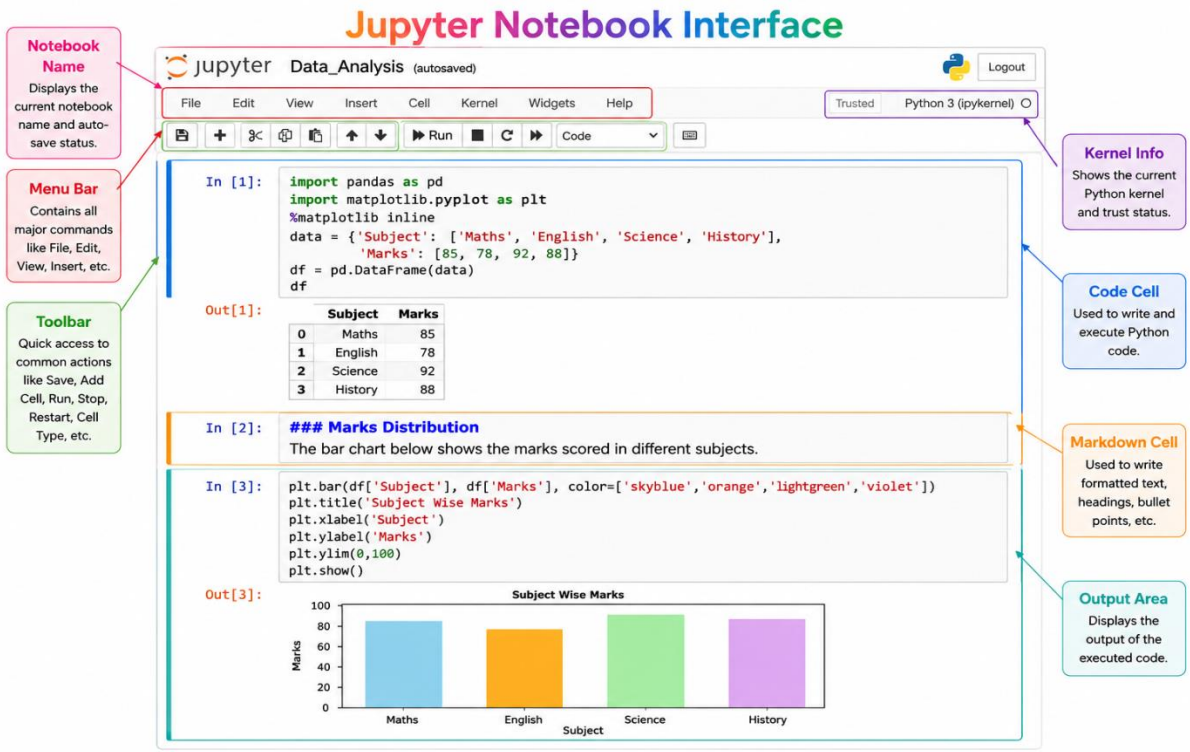
```
pip install notebook
```

Start Jupyter Notebook

Type:

```
jupyter notebook
```

It will open in your browser automatically.



Best for:

- Practice coding
- Learning Python
- Data analysis
- Small projects

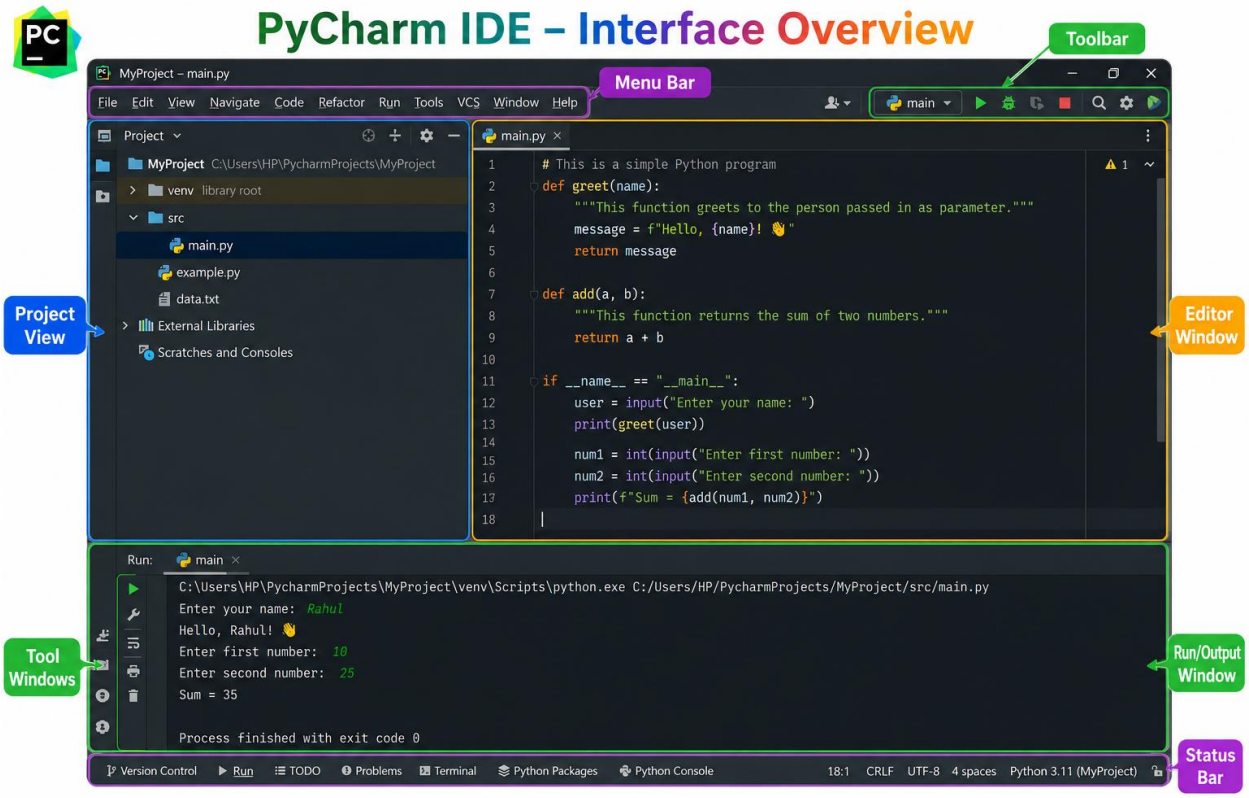
Option 2 — Install PyCharm

Download

Official website:

PyCharm Official Website

Choose: PyCharm Community Edition



Best for:

- Professional coding
- Large projects
- Debugging
- App development

Option 3 — Install VS Code

Download

Official website:

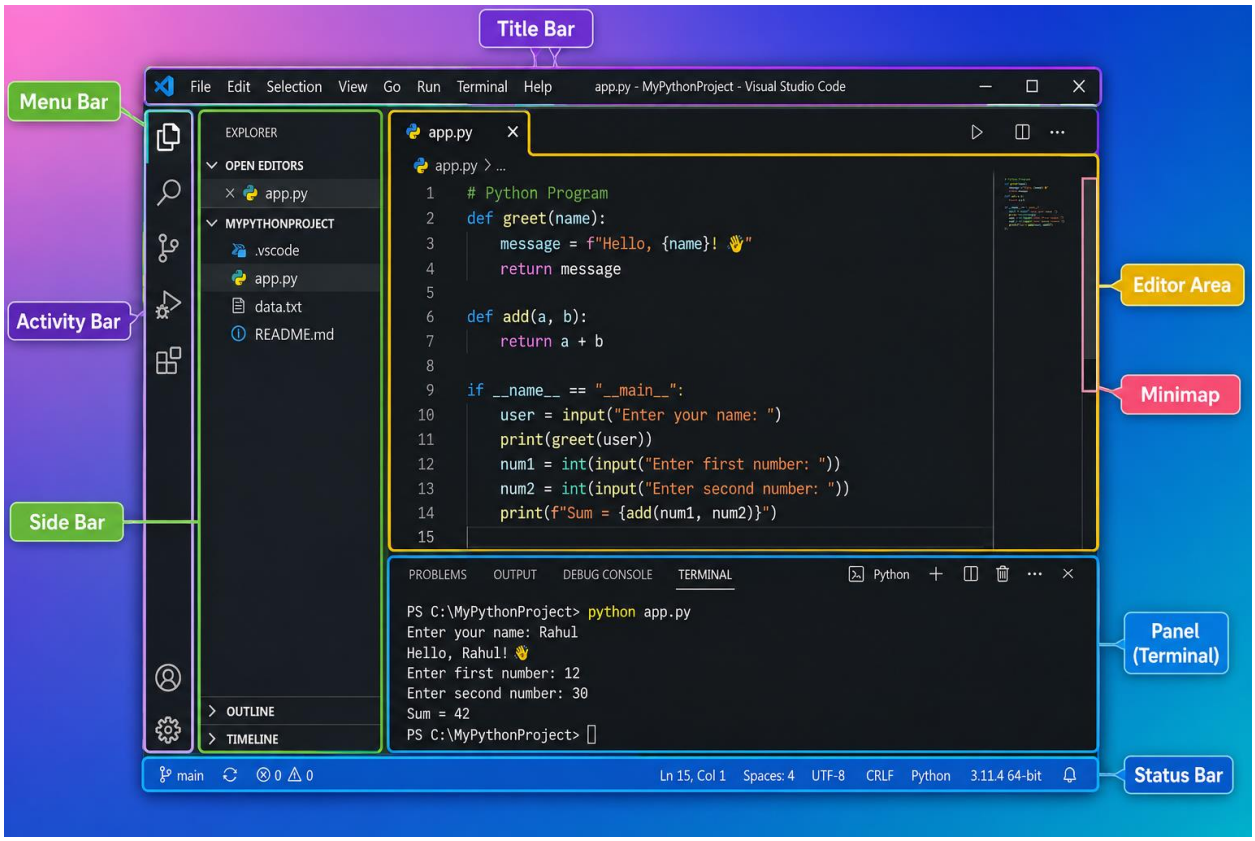
Install Python Extension

After installing VS Code:

1. Open VS Code

2. Click Extensions
3. Search:
4. Install extension by Microsoft

VS Code Interface



Best for:

- Web development
- Python + other languages
- Lightweight coding

Which One Should You Choose?

For absolute beginners:

✓Jupyter Notebook

For job-ready Python development:

✓PyChar

For all-purpose coding:

First Python Program

Create a file:

```
print("Hello World")
```

Run it and you will see:

```
Hello World
```

Python Uses

Python is widely used for:

- Web development
- Automation
- AI & Machine Learning
- Data analysis
- Cyber security
- Software development
- Freelancing projects on Fiverr and Upwork

An Integrated Development Environment (IDE)

An Integrated Development Environment (IDE) is a comprehensive software application that provide a text editor with essential development tools like compilers, debuggers, and project management features all in a single interface. While a simple text editor is lightweight and fast, an IDE is designed to handle the full software development lifecycle, especially for large or complex projects.

In simple words, an IDE is a program that helps developers write, test, and debug code more easily in one place.

Advantages of IDE over Text Editors

1. Code Completion (IntelliSense)

IDEs provide automatic suggestions for code, functions, and variables, which speeds up programming and reduces mistakes.

2.Built-in Debugging Tools

IDEs allow you to:

* Set breakpoints
* Execute code line by line
* Identify and fix errors easily

3. Error Detection

IDEs highlight syntax and logical errors instantly, helping programmers fix problems quickly.

4. Project Management

IDEs organize large projects with:

* File explorer
* Folder structure
* Multiple modules support

5. Built-in Terminal & Tools

You can run commands, scripts, and tests inside the IDE without switching applications.

6. Version Control Integration

IDEs support tools like Git, making it easier to:

* Save changes (commit)
* Track modifications
* Collaborate with teams

7. Code Formatting & Refactoring

IDEs automatically:

* Format code properly
* Rename variables safely
* Improve overall code structure

8**Support for Extensions and Plugins**

IDEs support plugins for:
* Web development
* Data science
* Database management
* AI/ML tools

9. Faster Development Because of automation and tools, development becomes faster and more efficient compared to text editors.

Python Variables and Data Types

Python Variables

Variables are defined with the assignment operator, "=". Python is dynamically typed, meaning that variables can be assigned without declaring their type, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function.

In programming, variables are fundamental building blocks that allow us to store, manipulate, and reuse data in our code. They are essential for creating dynamic and flexible programs.

A variable is like a labeled box where you can store data. Imagine you have a box labeled "age" and you put the number 25 in it. In Python, you would do this by writing:

```
age = 25
```

When you write this, Python creates a box (variable) with the name age and stores 25 in that box (variable).

Here's the cool part: whenever you use age in your code, Python will remember it is 25. For example:

```
print(age)# Output: 25
```

For example we can modify our Hello World program to ask the user for the name of their best friend and print out a welcome message to that best friend. If we want to, we can reuse the variable to hold the name of that best friend. For example:

```
print('Hello, world')
name = input('Enter your name: ')
print('Hello', name)
name = input('What is the name of your best friend: ')
print('Hello Best Friend', name)
```

When we run this version of the program and the user enters 'Sam' for their Name for their name and 'John' for their best friends' name we will see:

```
Hello, world
Enter your name: Sam
Hello John
What is the name of your best friend: Denise
Hello Best Friend Denise
```

Rules for Python Variable Names

When naming variables in Python, follow these rules:

1. Variable names can contain only letters, digits, and underscores (A-Z, a-z, 0-9, and _). Example: user_age, total_count2
2. A variable name must begin with a letter or an underscore, not with a digit. Valid: _count, name1
3. No spaces allowed. Valid: first_name Invalid: first name
4. Python variable names are case-sensitive, meaning age, Age, and AGE are treated as separate variables.
5. Avoid using Python keywords (like if, for, while) as variable names.

Types of Numbers

There are three types used to represent numbers in Python; these are integers (or integral) types, floating point numbers and complex numbers.

This begs the question why? Why have different ways of representing numbers; after all humans can easily work with the number 4 and the number 4.0 and don't need completely different approaches to writing them (apart from the '.' of course). This actually comes down to efficiency in terms of both the amount of memory needed to represent a number and the amount of processing power needed to work with that number. In essence integers are simpler to work with and can take up less memory than real numbers. Integers are whole numbers that do not need to have a fractional element. When two integers are added, multiplied or subtracted they will always generate another integer number.

In Python real numbers are represented as floating point numbers (or floats). These can contain a fractional part (the bit after the decimal point). Computers can best work with integers (actually of course only really 1s and 0s). They therefore

need a way to represent a floating point or real number. Typically this involves representing the digits before and after the decimal point.

Integers (int):

value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). There is no limit to how long an integer value can be. For Example:

```
</> python
x = 1
print(x)
print(type(x))
x =
10000000000000000000000000000000000000000000000000000000000000001
print(x)
print (type(x))
If this code is run then the output will show that both numbers are of type int:1
<class 'int'>
10000000000000000000000000000000000000000000000000000000000000001
<class 'int'>
```

This makes it very easy to work with integer numbers in Python. Unlike some programming languages such as C# and Java have different integer types depending on the size of the number, small numbers having to be converted into larger types in some situations.

Features

- Positive or negative numbers
- No decimal values

Floating-point numbers (float)

In Python, a floating-point number (or float) is a numeric data type used to represent real numbers that have a decimal point. Unlike integers, which are whole numbers, floats can include fractional parts and are essential for calculations requiring precision, such as scientific or financial data.

For Example:

```
</> python
exchange_rate = 1.83
print(exchange_rate)
print(type(exchange_rate))
This produces output indicating that we are storing the number 1.83 as a floating
point number:

1.83
<class 'float'>
```

Features

- Used for decimal values
- Can represent scientific numbers

String Type

In Python, a string means text made up of characters such as letters, numbers, symbols, and spaces. Strings in Python are surrounded by either single quotation marks, or double quotation marks. 'Hello' is the same as "Hello". Strings can be output to screen using the print function. For example: print("Hello").

```
print("It's the day")
print('She said "hello" to everyone')
```

The output of these two lines is:

```
It's the day
She said "hello" to everyone
```

Features

- Written inside single ' ' or double " " quotes
- Used to store text

Boolean Type

Python supports another type called Boolean; a Boolean type can only be one of True or False (and nothing else). Note that these values are True (with a capital T) and False (with a capital F); true and false in Python are not the same thing and have no meaning on their own.

Booleans represent True or False values.
The equivalent of the int or float class for Booleans is bool.The following example illustrates storing the two Boolean values into a variable.

```
</> python
all_ok:
all_ok = True
print(all_ok)
all_ok = False
print(all_ok)
print(type(all_ok))
The output of this is
True
False
<class 'bool'>
```

The Boolean type is actually a sub type of integer (but with only the values True and False) so it is easy to translate between the two, using the function int() and bool() to convert from Booleans to Integers and vice versa.

For example:

```
print(int(True))
print(int(False))
print(bool(1))
print(bool(0))

Which produces
1
0
True
False
```

Features

- Used in conditions and decision-making
- Only two values: True and False

Type casting

Type casting and operators are foundational concepts in programming. They allow you to manipulate data and build the logic needed to control how your programs make decisions and execute calculations.

1. Type Casting (Type Conversion)

Type casting is the process of converting a value from one data type to another. It helps ensure data is compatible for operations or formatted correctly.

Implicit Casting (Automatic): Handled safely by the compiler/interpreter. It typically happens when moving from a smaller data type to a larger one (e.g., converting an integer to a float) without losing precision. Python automatically converts one type to another when needed.

```
</> python
x = 5       # int
y = 2.5     # float

result = x + y
print(result)      # 7.5
print(type(result))  # float
```

```
  7.5
<class 'float'>
```

- **Explicit Casting (Manual):** Required when moving from a larger data type to a smaller one (e.g., float to integer). You must specify the target type in parentheses, which risks truncation or data loss.

Common functions:

- int() → converts to integer
- float() → converts to float
- str() → converts to string
- bool() → converts to Boolean

Examples:

Convert to integer

```
</> python
x = "20"
y = int(x)
print(y)           # 20
print(type(y))  # int
Running code
10
<class 'int'>
```

Convert to float

```
x = 5
y = float(x)
print(y)       # 5.0
Running code
5.0
<class 'float'>
```

Convert to String

```
x = 100
y = str(x)
print(y)       # "200"
print(type(y))  # str
Running code
200
<class 'str'>
```

Convert to Boolean

```
print(bool(0))     # False
print(bool(1))     # True
print(bool(""))    # False
print(bool("Hi"))  # True
Running code
False
True
False
True
```

Basic Operators

Operators are special symbols or keywords used to perform operations on variables and values.

Arithmetic Operators

These perform standard mathematical calculations.

- Addition: (+)
- Subtraction: (-)
- Multiplication: (*)
- Division: (/)

- Modulus: (%) (Returns the remainder of a division)
- Exponent (Power): (**)
- Floor Division (//)

Example of Arithmetic Operators

```
</> python
a = 12
b = 5

print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
print("Modulus:", a % b)
print("Power:", a ** b)
print("Floor Division:", a // b)




Running code
Addition: 17
Subtraction: 7
Multiplication: 60
Division: 2.4
Modulus: 2
Power: 248832
Floor Division: 2
```

Comparison (Relational) Operators

These compare two values and evaluate to a Boolean (True or False) result.

- Equal to: (==) (checks if values are the same)
- Not Equal to: (!=) or (<>)
- Greater than: (>)
- Less than: (<)
- Greater than or equal to: (>=)
- Less than or equal to: (<=)

Logical (Boolean) Operators

These combine multiple conditions or alter an existing Boolean value to control logic flow.

- Logical AND: && or and (Evaluates to True if both conditions are true)
- Logical OR: (Evaluates to True if at least one condition is true)
- Logical NOT: ! or not (Reverses the logical state; converts True to False and vice versa)

Input and output operations.

The print() function is used for output in various formats and the input() function enables interaction with users.

Python's input() function is used to take user input. By default, it returns the user input in form of a string.

```
name = input("Enter your name: ")

print("Hello,", name, "! Welcome!")
```

output operations

The print() function allows us to display text, variables and expressions on the console. In the below example, "Hello, World!" is a string literal enclosed within double quotes. When executed, this statement will output the text to the console.

```
print("Hello, World!")
```

Output

```
Hello, World!
```

Basic string operations and manipulation Techniques.

Basic string operations and manipulation Techniques.

In **Python**, strings are unchangeable collections of characters. Since strings are immutable, operations such as changing letter cases or replacing words do not alter the original string; instead, they generate a new string.

Main String Operations

These are the basic techniques used to work with strings without relying on advanced string functions.

- **Concatenation (+)**: Combines multiple strings into a single string.
  Example: 'Hello' + ' ' + 'World'
- **Repetition (*)**: Duplicates a string a certain number of times.
  Example: 'Hi' * 3 gives 'HiHiHi'
- **Indexing ([])**: Retrieves an individual character by its position.
  The first character starts at index 0, while -1 refers to the last character.
- **Slicing ([start:end])**: Obtains a section of a string.
  The starting index is included, but the ending index is excluded.
  Example: 'Python'[0:2] returns 'Py'

Manipulation Methods

Python provides a vast library of built-in string methods for cleaning and formatting text.

Category	Method	Description
Case Change	upper(), lower()	Converts all characters to uppercase or lowercase.
	title(), capitalize()	Capitalizes every word or just the first character.
Cleaning	strip()	Removes leading and trailing whitespace.
	replace(old, new)	Replaces occurrences of a substring with a different one.
Splitting/Joining	split(sep)	Breaks a string into a list based on a delimiter (default is space).
	join(iterable)	Combines a list of strings into one string using a separator.
Searching	find(sub)	Returns the lowest index of the substring or -1 if not found.
	startswith(), endswith()	Checks if the string begins or ends with a specific value.

Introduction to standard libraries

Python Standard Library is a set of built-in modules and packages that are automatically available with Python.It offers pre-defined functions for various tasks like file management, mathematical calculations, date and time handling, web operations, and more. The Python Standard Library is a massive collection of pre-written code (modules) that comes bundled with every Python installation.

Since these libraries come included with Python, there is no need for separate installation

Common Python Standard Libraries

Library	Purpose
math	Mathematical operations
random	Generate random numbers
datetime	Work with dates and time
os	Interact with operating system
sys	Access system-specific parameters
json	Handle JSON data
re	Regular expressions (pattern matching)
statistics	Statistical calculations

Introduction to Third-Party Libraries in Python

Third-party libraries in Python are external modules or packages created by developers outside the Python community.
These libraries are not included with Python by default and must be installed separately using tools like pip.

They provide additional features and functionalities for tasks such as data analysis, web development, machine learning, game development, and more.

Why Use Third-Party Libraries?

Third-party libraries help programmers to:

- Reduce coding effort
- Save development time
- Perform advanced tasks easily
- Build powerful applications efficiently

Popular Third-Party Libraries in Python

Library	Purpose
NumPy	Numerical and mathematical operations
Pandas	Data analysis and manipulation
Matplotlib	Data visualization and plotting
Requests	Sending HTTP requests
Flask	Web application development
TensorFlow	Machine learning and AI
OpenCV	Image and video processing

Installing and Uninstalling Libraries in Python

Python allows users to add extra functionality through libraries and packages.
These libraries can be easily installed or removed using the pip package manager.

Installing Libraries in Python

Libraries are installed using the pip install command.

Syntax

```
</>Python

pip install library_name

</>Python

pip install numpy
```

This command downloads and installs the NumPy library.

Installing a Specific Version

You can also install a particular version of a library.

```
</>Python

Pip list
```

Upgrading a Library

To update an existing library to the latest version:

```
</>Python

pip install --upgrade library_name
```

Example:

```
</>Python
pip install --upgrade matplotlib
```

Uninstalling Libraries in Python

Libraries can be removed using the pip uninstall command.

Syntax

```
</>Python
pip uninstall library_name
```

Example:

```
</> python
pip uninstall numpy
```

Python will ask for confirmation before uninstalling the library.

Multiple Choice Questions (MCQs) on Introduction to Python Programming

1. Python is a _____ language.

A) Low-level

B) High-level

C) Machine

D) Assembly

Answer: B) High-level

2. Who developed Python?

A) James Gosling

B) Dennis Ritchie

C) Guido van Rossum

D) Bjarne Stroustrup

Answer:C) Guido van Rossum

3. Which extension is used for Python files?

A) .java

B) .py

C) .cpp

D) .html

Answer: B) .py

4. Which of the following is an IDE for Python?

A) PyCharm

B) VS Code

C) Jupyter Notebook

D) All of these

Answer: D) All of these

5. Which command is used to install a Python library?

A) install

B) pip install

C) python install

D) add library

Answer: B) pip install

6. Which tool is mainly used for interactive coding and data science?

A) PyCharm

B) Jupyter Notebook

C) VS Code

D) Notepad

Answer: B) Jupyter Notebook

7. IDE stands for:

A) Integrated Development Environment

B) Internal Development Engine

C) Integrated Design Editor

D) Interface Development Environment

Answer: A) Integrated Development Environment

8. Which is an advantage of IDEs over text editors?

A) Debugging tools

B) Syntax highlighting

C) Auto-completion

D) All of these

Answer: D) All of these

9. A variable is used to:

A) Store data

B) Print output

C) Create loops

D) Install libraries

Answer: A) Store data

10. Which data type stores whole numbers?

A) float

B) string

C) int

D) bool

Answer: C) int

11. Which data type stores decimal values?

A) float

B) int

C) string

D) bool

Answer: A) float

12. Which data type stores True or False values?

A) string

B) bool

C) float

D) int

Answer: B) bool

13. Which symbol is used for comments in Python?

A) //

B) /* */

C) #

D) --

Answer:C) #

14. Which function is used to take input from the user?

A) print()

B) input()

C) read()

D) scan()

Answer: B) input()

15. Which function is used to display output?

A) show()

B) print()

C) display()

D) output()

Answer: B) print()

16. What is the output type of input() function?

A) int

B) float

C) string

D) bool

Answer:C) string

17. Type conversion from string to integer is done using:

A) str()

B) bool()

C) int()

D) float()

Answer: C) int()

18. Which operator is used for addition?

A) *

B) +

C) -

D) /

Answer: B) +

19. Which operator is used for exponentiation?

A) ^

B) //

C) **

D) %%

Answer: C)

20. Which operator gives remainder?

A) /

B) %

C) //

D) **

Answer: B) %

21. Which operator is used for logical AND?

A) &&

B) and

C) &

D) AND

Answer: B) and

22. Which operator checks equality?

A) =

B) !=

C) ==

D) <=

Answer: C) ==

23. What is the result of 5 > 3?

A) False

B) True

C) Error

D) None

Answer: B) True

24. Which operator is used for floor division?

A) /

B) %

C) //

D) **

Answer: C) //

25. Strings in Python are enclosed within:

A) Quotes

B) Brackets

C) Parentheses

D) Curly braces

Answer: A) Quotes

26. Which function converts data to float type?

A) str()

B) bool()

C) int()

D) float()

Answer: D) float()

27. Which of the following is a valid variable name?

A) 2name

B) user_name

C) class

D) user-name

Answer:B) user_name

28. Python is a:

A) Compiled language only

B) Interpreted language

C) Machine language

D) Markup language

Answer:B) Interpreted language

29. Which keyword represents Boolean false value?

A) false

B) FALSE

C) False

D) F

Answer: C) False

30. Which method converts a string to uppercase?

A) upper()

B) uppercase()

C) capital()

D) up()

Answer: A) upper()

31. Which method converts a string to lowercase?

A) lower()

B) lowercase()

C) small()

D) low()

Answer: A) lower()

32. Which method removes extra spaces from a string?

A) trim()

B) strip()

C) remove()

D) delete()

Answer: B) strip()

33. Which operator is used for string concatenation?

A) *

B) +

C) %

D) &

Answer: B) +

34. What is the output of len("Python")?

A) 5

B) 6

C) 7

D) Error

Answer: B) 6

35. Which library is used for mathematical operations?

A) os

B) math

C) random

D) sys

Answer: B) math

36. Which library is used for random number generation?

A) math

B) random

C) json

D) os

Answer: B) random

37. Which library is used for working with dates and time?

A) datetime

B) date

C) calendar

D) clock

Answer: A) datetime

38. Which library helps interact with the operating system?

A) random

B) os

C) sys

D) math

Answer: B) os

39. Libraries included with Python are called:

A) Third-party libraries

B) Standard libraries

C) External libraries

D) Custom libraries

Answer: B) Standard libraries

40. Which package manager is used in Python?

A) npm

B) pip

C) gem

D) apt

Answer: B) pip

41. Which command lists installed libraries?

A) pip show

B) pip list

C) pip libraries

D) pip check

Answer: B) pip list

42. Which command removes a library?

A) pip remove

B) pip delete

C) pip uninstall

D) pip erase

Answer: C) pip uninstall

43. Which of the following is a third-party library?

A) math

B) os

C) NumPy

D) sys

Answer: C) NumPy

44. Which library is used for data analysis?

A) Pandas

B) random

C) os

D) math

Answer: A) Pandas

45. Which library is commonly used for machine learning?

A) TensorFlow

B) os

C) sys

D) datetime

Answer: A) TensorFlow

46. Which symbol is used for multiplication?

A) x

B) *

C) %

D) #

Answer: B) *

47. Which of the following is NOT a Python data type?

A) int

B) float

C) char

D) bool

Answer: C) char

48. Which method finds the position of a substring?

A) search()

B) locate()

C) find()

D) indexof()

Answer: C) find()

49. Which statement is correct about Python?

A) Python is case-sensitive

B) Python ignores indentation

C) Variables must be declared first

D) Python uses semicolons compulsorily

Answer: A) Python is case-sensitive

50. Which of the following is used to upgrade a library?

A) pip upgrade

B) pip install --upgrade

C) pip update

D) pip latest

Answer: B) pip install –upgrade

Short Questions Answers

1. What is Python?

Python is a high-level, easy-to-learn programming language used for web development, data science, AI, automation, and software development.

2. What is an IDE?

An IDE (Integrated Development Environment) is software used to write, run, and debug programs. Examples: PyCharm, Visual Studio Code, Jupyter Notebook.

3. Advantages of IDEs over Text Editors.

* Syntax highlighting

* Auto-completion

* Error detection

* Easy debugging

* Faster coding and execution

4. What are Variables in Python?

Variables are used to store data values.

Example:

```
</> python

x = 10

name = "Ram"
```

5. Explain Integer Data Type

Integer (`int`) stores whole numbers without decimal points.

Example:

```
</> python

a = 25
```

6. Explain Float Data Type

Float stores decimal numbers.

Example:

```
</> python

price = 99.5
```

7. Explain String Data Type

String (`str`) stores text or characters inside quotes.

Example:

```
</> python

name = "Python
```

8. Explain Boolean Data Type

Boolean stores only two values: `True` or `False`.

Example:

```
python

is_pass = True
```

9. What is Type Casting?

Type casting means converting one data type into another.

Example:

```
</> python

x = int("5")
```

10. What are Arithmetic Operators?

Arithmetic operators perform mathematical calculations.

Examples: +, -, *, /, %

11. What are Comparison Operators?

Comparison operators compare two values.

Examples: ==, !=, >, <

12. What are Logical Operators?

Logical operators combine conditions.

Examples: and, or, not

13. Explain Input Operation in Python

input() is used to take input from the user.

Example:

```
</> python

name = input("Enter name:")
```

14. Explain Output Operation in Python

print() is used to display output.

Example:

```
</>python

print("Hello")
```

15. What are Basic String Operations?

Basic string operations include:

* Concatenation (+)

* Repetition (*)

* Length using len()

16. What is String Manipulation?

String manipulation means modifying strings using methods like:

* upper()

* lower()

* replace()

* split()

17. What are Standard Libraries?

Standard libraries are built into Python and available by default.

Example: math, random, os

18. What are Third-Party Libraries?

Third-party libraries are external packages installed by users.

Examples: NumPy, Pandas

19. How to Install a Python Library?

Use pip install command.

Example:

```
</> bash

pip install numpy
```

20. How to Uninstall a Python Library?

Use `pip uninstall` command. Example:

```
</> bash

pip uninstall numpy
```

Long Questions:

1. Explain the Installation of Python and IDEs.
2. Explain Python Variables and Data Types.
3. Explain Type Casting and Basic Operators in Python.
4. Explain Input and Output Operations in Python.
5. Explain Basic String Operations and String Manipulation Techniques.
6. Explain Standard Libraries and Third-Party Libraries in Python.
7. Explain arithmetic, comparison, and logical operators in Python.

Case-Based Learning and Problem-Based Learning in Python

1. Case-Based Learning: Installing Python and IDE

Case Study

A student wants to learn Python programming for data science but cannot write or run Python programs on the computer.

Problem

The student does not have Python or any IDE installed.

Solution

Step 1: Install Python

1. Visit Python Official Website
2. Download Python.
3. Run installer and select “Add Python to PATH”.
4. Complete installation.

Step 2: Install IDE

The student can use:

- PyCharm
- Visual Studio Code
- Jupyter Notebook

Advantages of IDEs

- Automatic error checking
- Easy debugging
- Syntax highlighting
- Faster coding

Learning Outcome

The student can now write, run, and debug Python programs easily.

2. Problem-Based Learning: Variables and Data Types

Problem

A shopkeeper wants to store product information such as:

- Product name
- Price
- Quantity
- Availability

Solution Using Python

```
</> python
product_name = "Laptop"
price = 55000.50
quantity = 10
available = True

print(product_name)
print(price)
print(quantity)
print(available)
```

Concepts Used

- String → Product name
- Float → Price
- Integer → Quantity
- Boolean → Availability

Learning Outcome

Students learn how different data types store different kinds of information.

3. Case-Based Learning: Type Casting

Case Study

A teacher collects student marks using keyboard input. Input values are stored as strings, but calculations require integers.

Problem

Addition cannot be performed correctly on string values.

Solution

```
</> python
marks1 = input("Enter marks: ")
marks2 = input("Enter marks: ")

total = int(marks1) + int(marks2)

print("Total Marks:", total)
```

Learning Outcome

Students understand how type casting converts string data into integers for calculations.

4. Problem-Based Learning: Operators

Problem

A company wants a program to calculate employee salary bonus.

Solution

```
</> python
salary = 50000
bonus = salary * 10 / 100

print("Bonus:", bonus)
```

Operators Used

- Arithmetic Operator → *, /
- Comparison Operator Example:

```
</> python>
print(salary > 40000)
```

- Logical Operator Example:

```
</> python
print(salary > 30000 and salary < 60000)
```

Learning Outcome

Students learn practical use of operators in real-life calculations.