Basics of Python Programming for Pharmaceutical Sciences

BOOKS FOR BACHELOR OF PHARMACY

- Basics of Python Programming for Pharmaceutical Sciences (Theory)
- General Pharmacy (Theory)
- Healthcare Psychology and Communication Skills (Theory)
- Human Anatomy, Physiology and Pathophysiology I (Theory)
- Introduction to Pharmacognosy (Theory)
- Pharmaceutical Inorganic and Analytical Chemistry (Theory)
- Applied Biostatistics and Data Analytics for Pharmaceutical Sciences (Theory)
- Biochemistry (Theory)
- Human Anatomy, Physiology and Pathophysiology II (Theory)
- Pharmaceutical Organic Chemistry (Theory)
- Pharmacognosy and Phytochemistry (Theory)
- Physical Pharmaceutics (Theory)

Theory & Practical of All Books are available

HIGHLIGHTS OF THE BOOK

- Simple and student-friendly language for easy understanding of Python concepts and applications.
- Well-structured content with easy Python Programs, detailed explanation of programs and outputs.
- Includes Practice Questions covering MCQs with answers, Short answer questions
- Long answer Questions and Programming Practice for effective learning
- Designed for Classroom Teaching, Laboratory Work and Self-Programming Practice.
- Practical Learning combining theory, examples and hands-on exercises.
- Comprehensive Coverage of Programming with Data Analysis and Visualization in a single resource.
- Real-World Case Studies and application-based examples of Pharmaceutical domain to connect learning with practice.
- Develops Programming, Analytical and Problem-Solving Skills in easy way.

9 789395 685054

SHREE SAI PRAKASHAN
407/2, New Mohanpuri, Meerut-250002

ShreeSai's®

Basics of Python Programming for Pharmaceutical Sciences

Er. Mahendra K. Verma

ShreeSai's®

Er. Mahendra K. Verma

Basics of Python Programming for Pharmaceutical Sciences

(STRICTLY ACCORDING TO SYLLABUS APPROVED BY PCI FOR B.PHARMA COURSE)

For B. Pharm
1st Sem

SHREE SAI PRAKASHAN

UNIT I

Introduction to Python Programming

CONTENTS

- Installing Python and an Integrated Development Environment (IDE)
  - *Jupyter Notebook, PyCharm, VS Code etc.,*
- Advantages of IDEs over text editors.
- Python variables,
  - *Python data types*
  - *integers,*
  - *floats,*
  - *strings,*
  - *booleans*
- Type casting
- Basic Operators
  - *Arithmetic,*
  - *Comparison,*
  - *Logical.*
- Input and output operations.
- Basic string operations and Manipulation Techniques.
- Introduction to standard libraries
  - *Third-party libraries,*
  - *Installing and Uninstalling libraries.*

About the Python Programming

Python is a high-level, interpreted, general-purpose, easy-to-understand Object Oriented Programming Language developed by "Guido van Rossum" and first released in 1991. It is used to create software, websites, automation scripts, data analysis tools, Artificial Intelligence and Machine Learning applications, and more. It's simple to use, packed with features and supported by a wide range of libraries and frameworks. Its clean syntax makes it beginner-friendly.

Why Python is Popular?

- **Simple syntax** - easy for beginners to learn.
- **Versatile** - used in web development, AI, machine learning, data science, automation, and game development.

- **Large community** - many libraries, tutorials, and resources are available.
- **Cross-platform** - runs on Windows, macOS, Linux, and other systems.

What can you Develop Using Python?

Python is one of the most versatile programming languages. You can develop applications ranging from simple scripts to advanced AI systems. Here are the major categories:

1. Desktop Applications

Create software that runs on Windows, macOS, and Linux.

Examples :

- Calculator
- Text Editor
- Student Management System
- Inventory Management System
- Media Player
- Library Management System

2. Web Applications

Develop dynamic websites and web portals.

Examples :

- College Website
- E-commerce Website
- Hospital Management System
- Learning Management System (LMS)
- Blog Platform
- Online Examination System

3. Mobile Applications

Develop Android and iOS applications.

Examples :

- To-Do App
- Attendance App
- Weather App
- College App

4. Artificial Intelligence (AI)

Develop intelligent systems that can learn and make decisions.

Examples :

- Chatbots
- Recommendation Systems
- Image Recognition
- Speech Recognition
- AI Tutors
- Virtual Assistants

5. Machine Learning

Build predictive models using data.

Examples :

- House Price Prediction
- Student Performance Prediction
- Disease Prediction
- Sales Forecasting
- Spam Detection

6. Data Science and Analytics

Analyze and visualize large datasets.

Examples :

- Business Dashboards
- Sales Reports
- Financial Analysis
- Survey Analysis

7. Data Visualization

Create interactive charts and graphs.

Examples :

- Bar Charts
- Pie Charts
- Line Graphs
- Heat Maps
- Dashboards

8. Automation

Automate repetitive tasks.

Examples :

- Rename Files
- Send Emails Automatically
- Excel Report Generation
- PDF Processing
- Data Entry Automation

9. Web Scraping

Collect information from websites.

Examples :

- Price Comparison
- News Collection
- Job Listings
- Product Information

10. Game Development

Develop 2D and simple 3D games.

Examples :

- Snake Game
- Tic-Tac-Toe
- Chess
- Space Shooter
- Platform Games

11. Internet of Things (IoT)

Develop applications for smart devices.

Examples :

- Smart Home
- Smart Agriculture
- Weather Monitoring
- Smart Irrigation
- Smart Parking

12. Robotics

Control robots and automate robotic systems.

Examples :

- Line Following Robot
- Obstacle Avoidance Robot

- Robotic Arm
- Drone Programming

13. Cybersecurity Tools

Develop security and network analysis tools.

Examples :

- Password Generator
- Port Scanner
- Packet Analyzer
- Vulnerability Scanner

14. Computer Vision

Process and analyze images and videos.

Examples :

- Face Detection
- Number Plate Recognition
- Object Detection
- Image Classification

15. Database Applications

Develop applications that store and manage data.

Examples :

- Employee Database
- Student Database
- Banking System
- Library System

16. Cloud Applications

Build cloud-based services and APIs.

Examples :

- File Storage
- Cloud Backup
- Server Monitoring
- REST APIs

17. Scientific Computing

Perform mathematical and engineering computations.

Examples :

- Numerical Simulation
- Signal Processing
- Statistical Analysis
- Engineering Calculations

18. Natural Language Processing (NLP)

Enable computers to understand human language.

Examples :

- Language Translation
- Sentiment Analysis
- Text Summarization
- Question Answering

19. APIs and Micro Services

Develop backend services for applications.

Examples :

- Weather API
- Student API
- Payment API
- Authentication Service

20. Command-Line Tools

Develop utilities that run in the terminal.

Examples :

- File Organizer
- Backup Tool
- Password Manager
- Log Analyzer

Features of Python

- Simple and Easy to Learn
- Interpreted Language
- High-Level Language
- Object-Oriented
- Portable
- Open Source
- Large Standard Library

- Dynamically Typed
- Supports Multiple Programming Paradigms
- Extensible
- Database Connectivity Support
- GUI Support
- Automatic Memory Management
- Large Community Support

Importance of Python in Pharmaceutical Sciences

Python is a high-level programming language widely used in pharmaceutical sciences because of its simplicity, flexibility, and powerful data-handling capabilities. It helps researchers and healthcare professionals manage, analyze, and interpret complex pharmaceutical and clinical data efficiently. Python is commonly applied in areas such as drug discovery, patient data analysis, machine learning, bioinformatics, and simulation of pharmaceutical compounds, supporting the development of safer and more effective medicines.

Why Python is important for a Pharmacist?

Learning Python is valuable for pharmacists because it helps improve workflow efficiency, patient care, and research activities. Python can automate repetitive tasks such as data entry, analyze patient data to identify trends and predict outcomes, and simulate clinical or pharmaceutical scenarios to support better decision-making and medication management.

1.1 Installing Python and an Integrated Development Environment (IDE)

Installation of Jupyter Notebook

Jupyter Notebook is an an open-source web application that provide interactive computing environment to the users to write, execute, and document the code in a single web-based document, making it a powerful tool for programming, data analysis, and scientific research.

It is widely used for Python programming, data analysis, machine learning, scientific computing, and education.

Note

- *Anaconda is an open-source distribution of the Python programming languages designed for Data Science, Machine Learning, and Scientific Computing.*
- *It simplifies Package Management and Software Deployment by providing a complete environment with numerous Pre-Installed Libraries and Tools, including Jupyter Notebook, making it easier to develop and manage data-driven projects.*

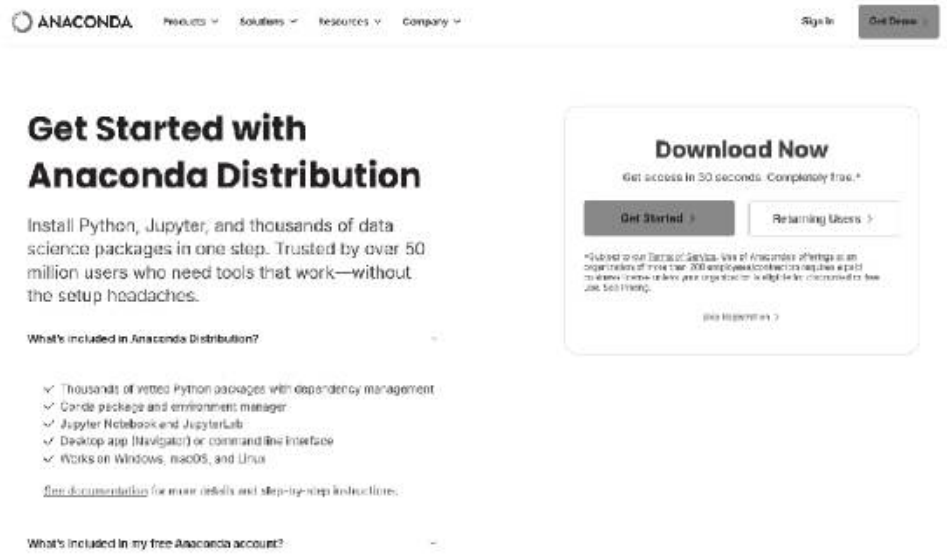
Remember

- *Jupyter Notebook is included as a pre-installed application in the Anaconda distribution. This means you do not need to install Jupyter Notebook separately. After installing Anaconda, you can launch Jupyter Notebook directly through the Anaconda Navigator graphical interface.*

Step to Install Jupyter Notebook on Windows

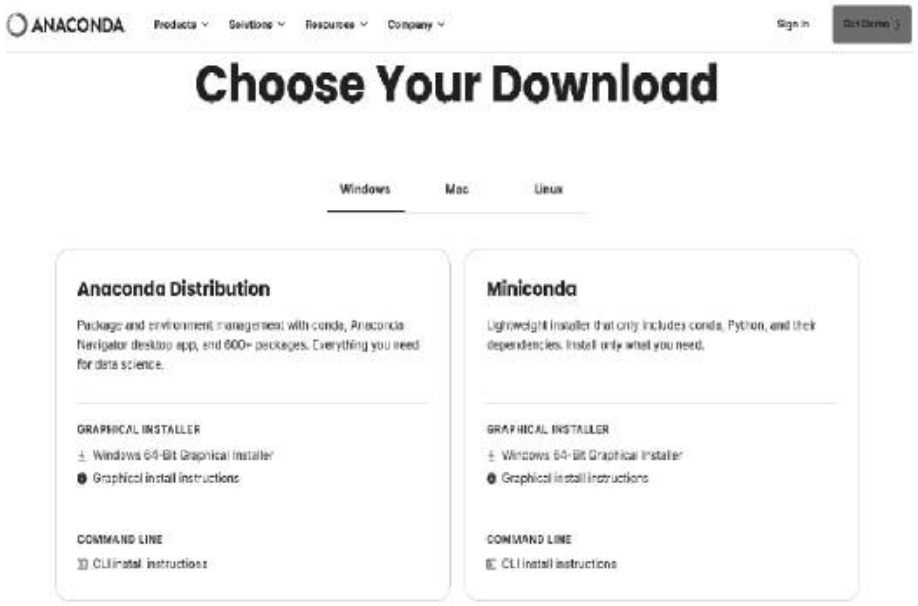
Step 1. Download Anaconda

- Go to the official website: https://www.anaconda.com/downloads
- Click on "Skip Registration" for free version for Windows.
- Select the latest version of Python 3.x.

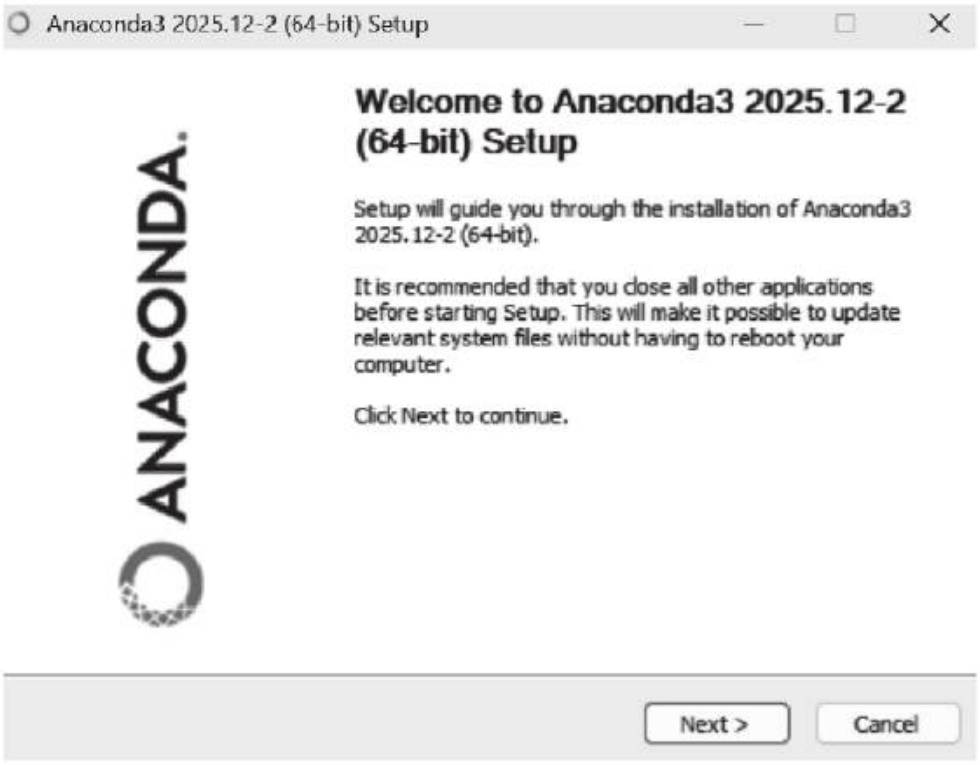


Step 2. When you click on "Skip Registration" you will be directed to Graphical Installer of Anaconda Distribution. Just download it by clicking on Windows 64-Bit Graphical Installer. After that it will start downloading.

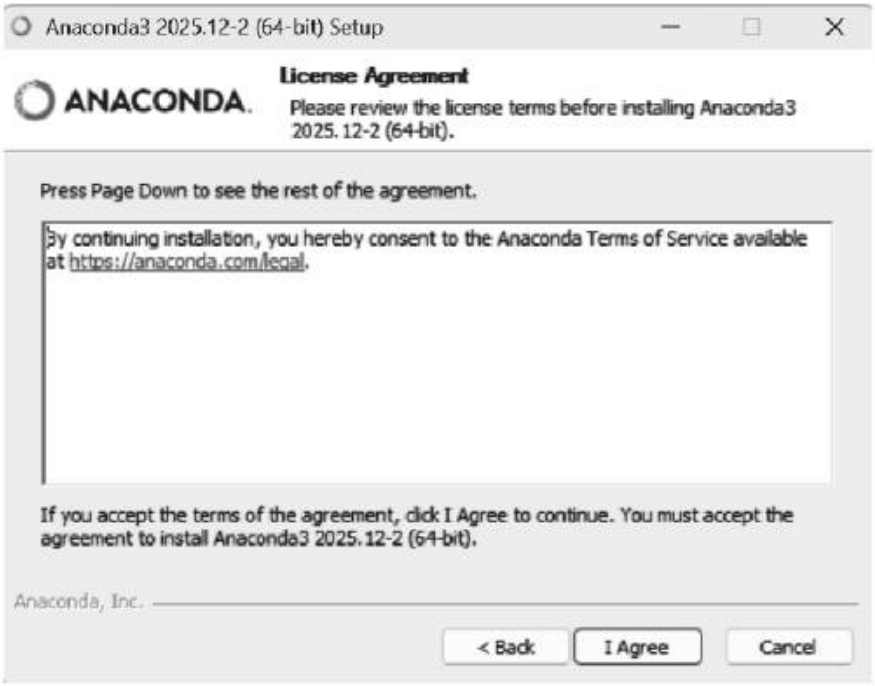
Step 3. Double click the installer to launch.



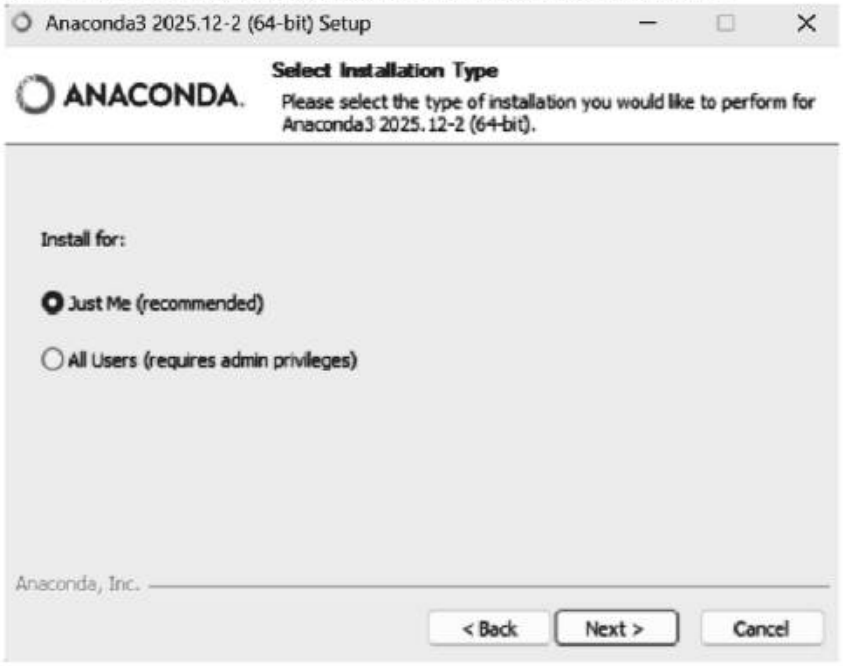
Step 4. Click Next.



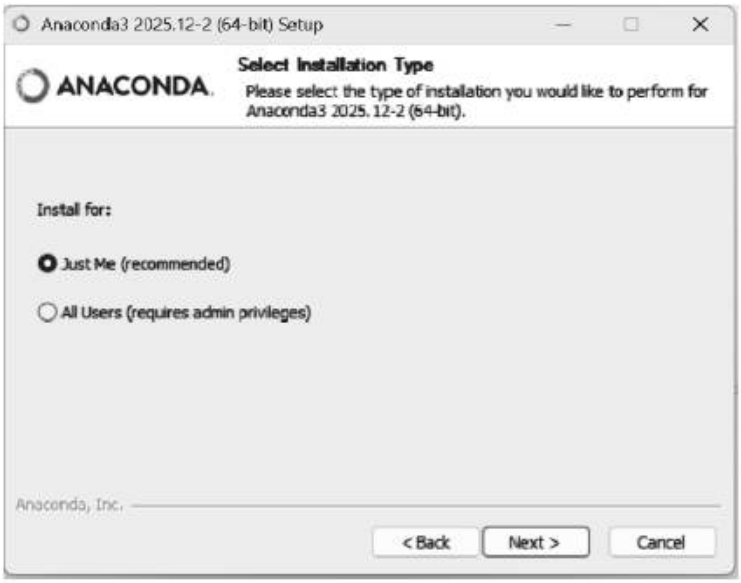
Step 5. Read the license Agreement and then click **"I Agree".**



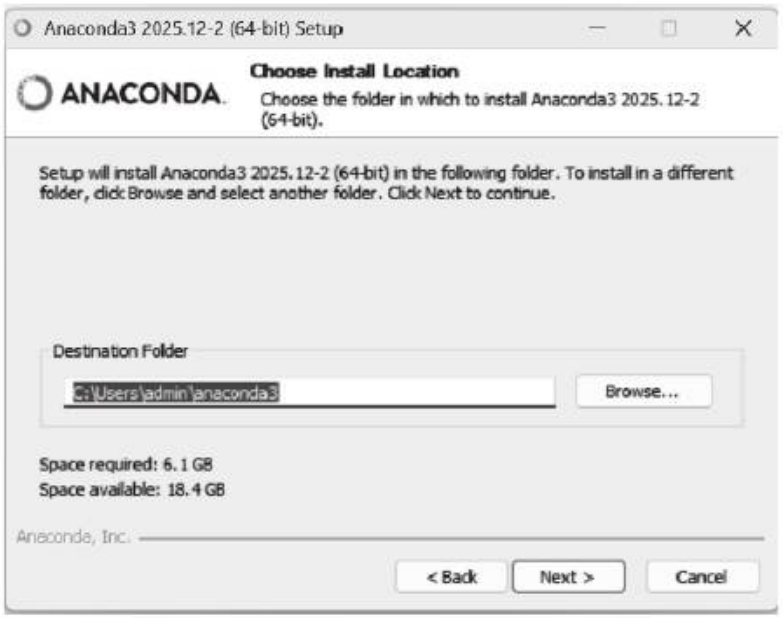
Step 6. Read the license Agreement and then click **"I Agree"**.



Step 7. Now select the Installation Type **"Just Me"** unless you're installing for all users (which require Windows Administrator privileges) and click Next.



Step 8. Select a **destination folder** to install Anaconda and click the **Next button.**

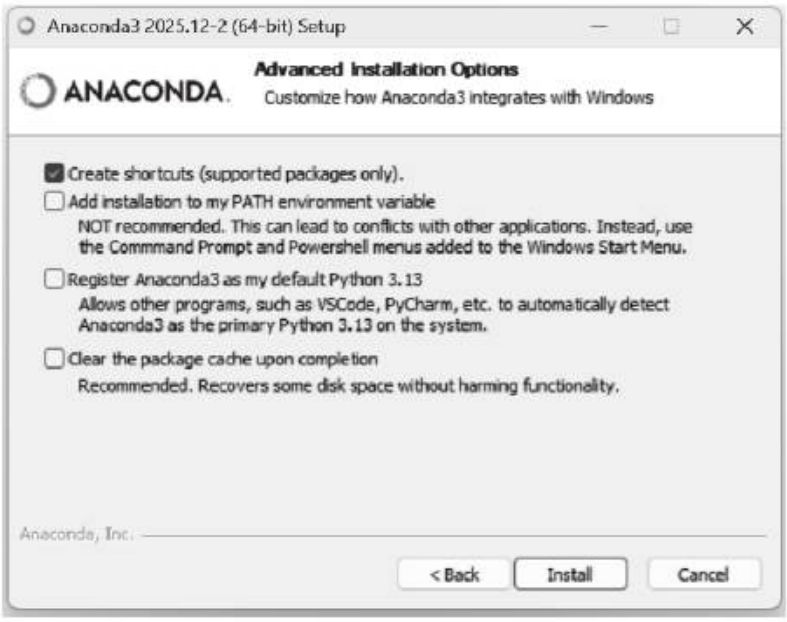


Step 9. Select option "add installation to my PATH environment variable". It is recommended that you should not add Anaconda to the PATH environment variable, because this may interfere with some other software.

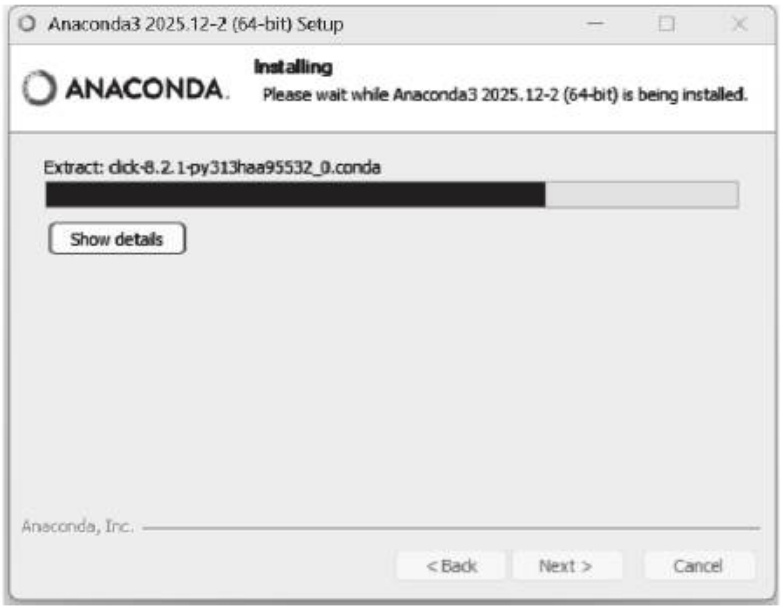
Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.

NOTE

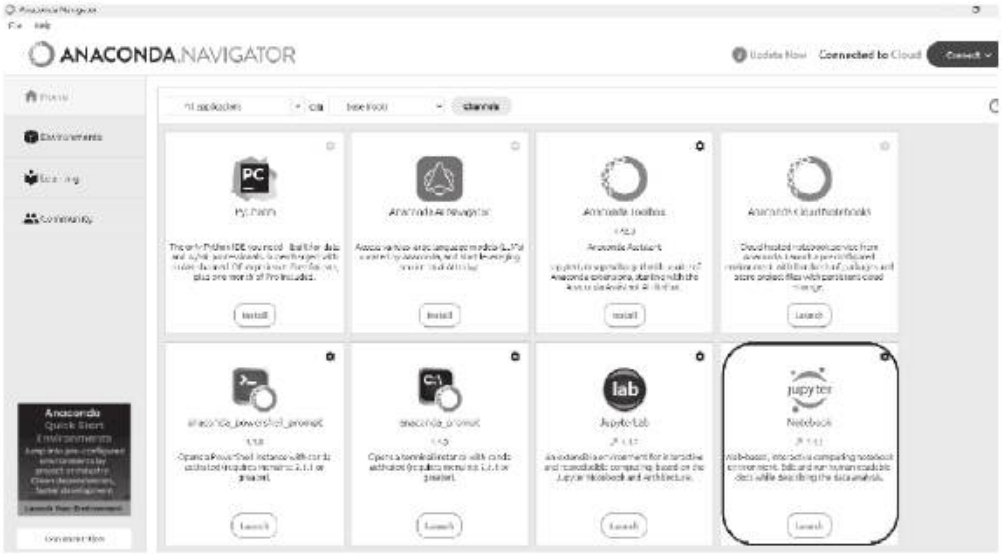
- *You can go for the option to register Anaconda as your default Python. Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, accept the default and leave this box checked.*



Step 10. After clicking on the Install button, Anaconda will be installed successfully.



When Anaconda Navigator is launched, you will see the icon of **"jupyter Notebook"**, as shown below:



Also you can follow the given steps:

After installation :

- Open the Start Menu.
- Search for "Anaconda Navigator" and open it.
- In Anaconda Navigator, click "Launch" under Jupyter Notebook.

Jupyter Notebook will open automatically in your default web browser.

Installation of PyCharm

PyCharm is a popular a cross-platform Integrated Development Environment (IDE) for Python development made by Jet Brains. It provides tools that make writing, testing, debugging, and managing Python code much easier than using a simple text editor.

You can download PyCharm Community Edition and proceed with the installation as given below:

Step 1. Download PyCharm Community Edition

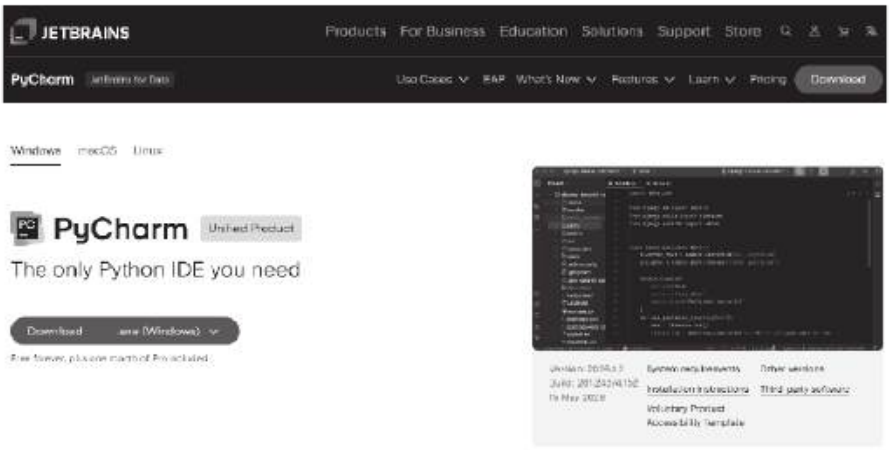
(i) Open your web browser.

(ii) Go to the official PyCharm website :

https://www.jetbrains.com/pycharm/download/?section=windows

(iii) Select Community Edition (Free).

(iv) Click Download.

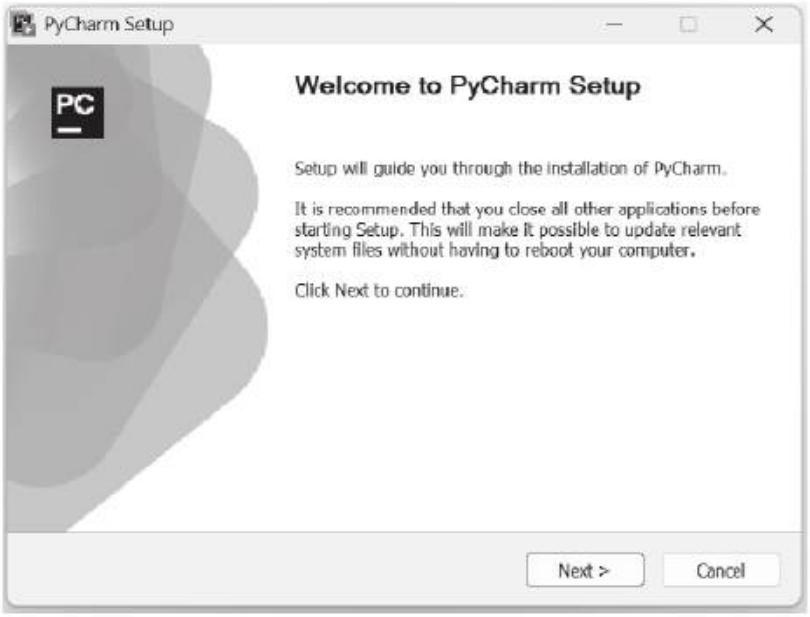


Step 2. Run the Installer

(i) After downloading, locate the installer file in your Downloads folder.

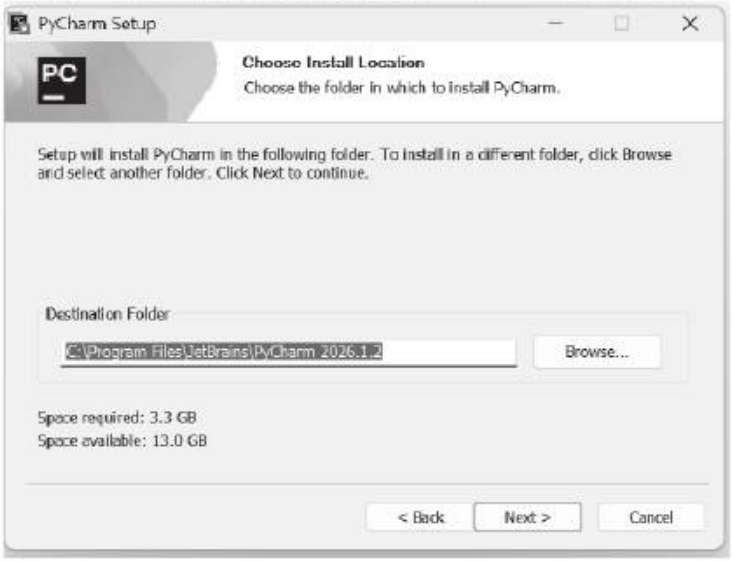
(ii) Double-click the installer file.

(iii) Click Next to start the installation.

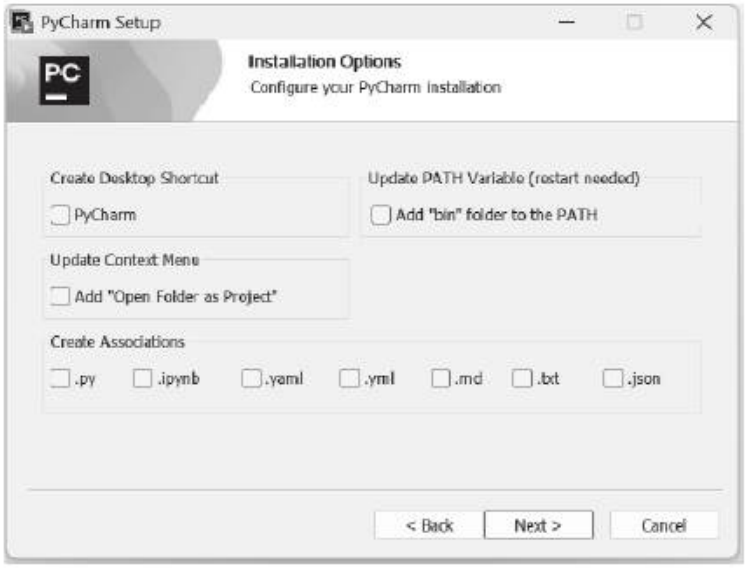


Step 3. Choose Installation Options

1. Select the installation location (or keep the default location).



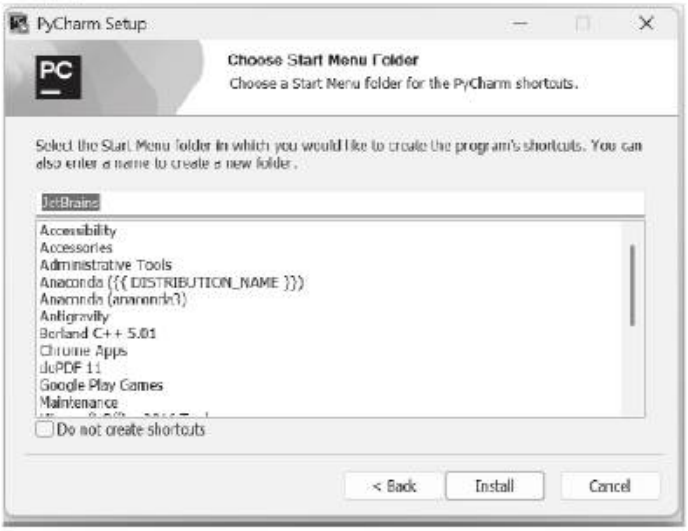
2. You may check the following options:
   - Create Desktop Shortcut
   - Add "Open Folder as Project"
3. Click Next.

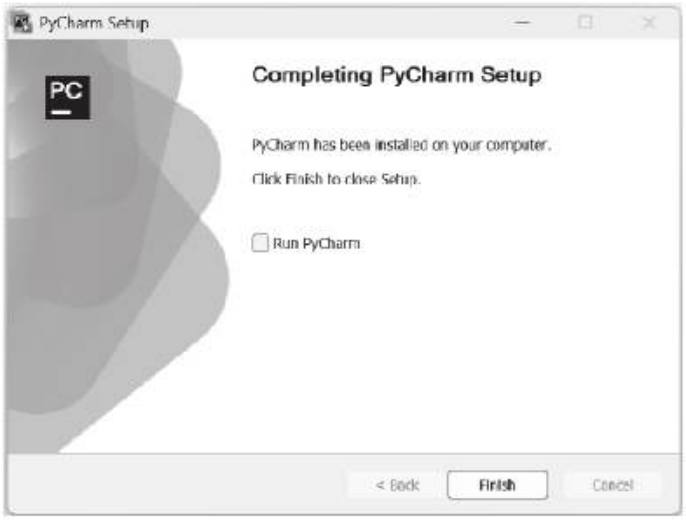


Step 4. Install PyCharm

1. Click Install.

2. Wait for the installation process to complete.
3. Click Finish.





Step 5. Launch PyCharm

1. Open PyCharm Community Edition from the desktop shortcut or Start Menu.
2. Accept the license agreement if prompted.

Installation of Vs Code

Visual Studio Code (VS Code) is a free, light weight, and powerful Source-Code Editordeveloped by Microsoft. It is widely used for writing, editing, debugging, and

running programs in many programming languages, including Python, Java, C++, JavaScript, and more.

Step 1. Download VS Code

1. Open your web browser.
2. Visit the official website:
   https://code.visualstudio.com/download

3. Click **Download for Windows** (or choose the version for your operating system).

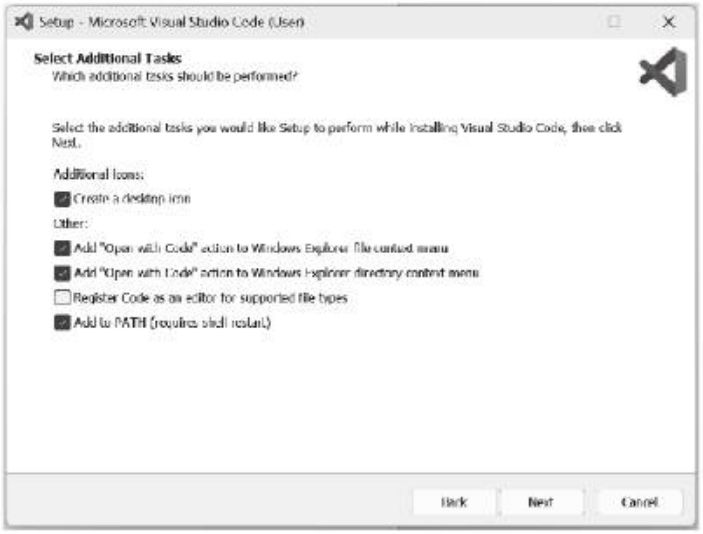
Step 2. Run the Installer

1. Open the downloaded installer file.

2. Click Next.
3. Accept the license agreement and click Next.

Step 3. Select Additional Tasks

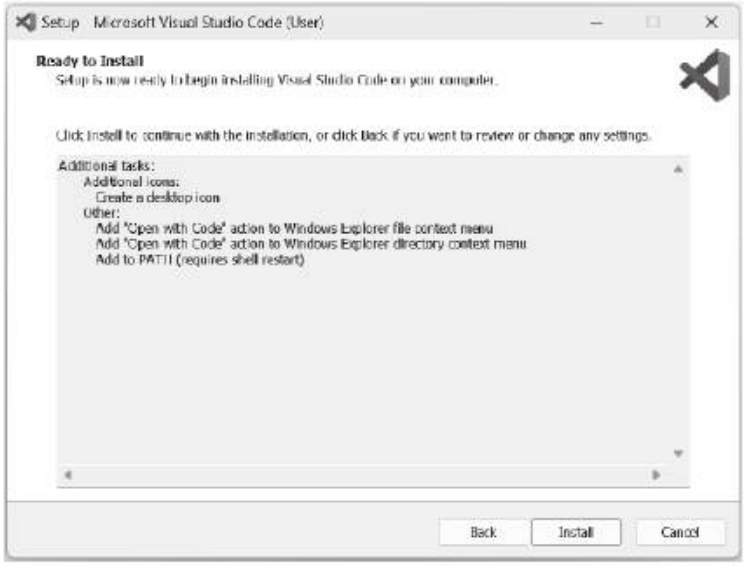
1. It is recommended to check the following options:
2. Create a desktop icon
3. Add "Open with Code" to Windows Explorer
4. Register Code as an editor for supported file types
5. Add VS Code to **PATH**



Click Next.

Step 4. Install VS Code

1. Click Install.



2. Wait for the installation to complete.
3. Click Finish.

Step 5. Launch VS Code

1. Open VS Code from the desktop shortcut or Start Menu.
2. The VS Code editor window will appear.

1.2 Advantages of IDEs Over Text Editors

IDEs stands for Integrated Development Environment. An IDEs is a software application used to write, run, and test programs easily.It provides all the tools needed for programming at one place.

Remember

- *An IDEs is a software tool that helps programmers to write and execute programs easily."*

Main Features of an IDEs

Important Features of an IDEs are:

1. Writing code,
2. Running programs,
3. Finding and correcting errors (debugging),
4. Auto code suggestions,
5. Managing files and projects etc.

Examples of IDEs for Python Programming

Examples of IDEs for Python Programming are:

1. PyCharm,
2. Jupyter Notebook,
3. Visual Studio Code etc.

Advantages of IDEs Over Text Editors

Important Advantages of IDEs over text editors are:

1. **Auto Code Completion**

   IDEs suggest code while typing, which saves time and reduces errors.

2. **Error Detection and Highlighting**

   Mistakes in code are identified automatically with warnings or highlights.

3. **Easy Debugging**

   IDEs provide debugging tools to find and fix errors step-by-step.

4. **Run Code Directly**
   Python programs can be executed directly inside the IDE.
5. **Better Project Management**
   IDE help organize files, folders, and projects efficiently.
6. **Syntax Highlighting**
   Different colors are used for keywords, variables, and strings, making code easier to read.
7. **Integrated Terminal and Tools**
   Many IDE include built-in terminals and package management support.
8. **Improved Productivity**
   Features like shortcuts, templates, and navigation make coding faster and easier.

1.3 Python Variables

Variables

In Python, a variable is basically a name that is assigned to a value. These variables are used to store data values.

Creating Variables in Python

In Python, you do not need to declare the data type of a variable before assigning a value to it. Python follows dynamic typing, which means the variable's type is decided automatically at runtime according to the value assigned. This makes the syntax simpler and the code more flexible.

Remember *

- *Python is a completely Object-Oriented Programming Language.*
- *Python is a Dynamically Typed Language, i.e. there is no need to declare variables or specify their data types before using them, as every variable is treated as an object.*

A variable is created when you assign a value to it.A variable can be created by simply choosing a name and assigning a value using the assignment operator (=).

Program

```
var1 = 5
var2 = "Python Programming"
print(var1)
print(var2)
```

Output

```
5
Python Programming
```

Explanation

In this program, two variables are created and assigned different types of values.

- `var1 = 5`
  Stores the integer value 5 in the variable `var1`.
- `var2 = "Python Programming"`
  Stores the string value "Python Programming" in the variable `var2`.
- `print(var1)`
  Displays the value stored in `var1`.
- `print(var2)`
  Displays the value stored in `var2`.

The output shows the values of both variables:

Rules for Naming Variables in Python

Rule 1: Variable names must start with a letter or underscore (_).

Program

```
Subject = "Python Programming"
_marks = 19
print(Subject)
print(_marks)
```

Output

```
5
Python Programming
```

Rule 2: A variable name cannot start with a number.

Program

```
1stSubject = "Python Programming"
print(1stSubject)
```

Output

```
  Cell In[2], line 1
    1stSubject = "Python Programming"
    ^
SyntaxError: invalid decimal literal
```

Fix Code

Rule 3: Variable names can contain letters, numbers, and underscores.

Program

```
Subject_1 = "Python Programming"
Subject_2 = "Java Programming"
print(Subject_1)
print(Subject_2)
```

Output

```
Python Programming
Java Programming
```

Rule 4: Special characters and spaces are not allowed.

Program

```
Subject-Name = "Python Programming"
MST Marks = 19
print(Subject-Name)
print(MST Marks)
```

Output

```
  Cell In[6], line 1
    Subject-Name = "Python Programming"
    ^
SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of '='?
```

Fix Code

Rule 5: Python keywords cannot be used as variable names.

Program

```
def=5
print(def)
```

Output

```
  Cell In[8], line 1
    class=5
          ^
SyntaxError: invalid syntax
```

Fix Code

Rule 6: Variable names are case-sensitive.

Program

```
SUBJECT = "Python Programming"
subject = "Java Programming"
print(subject)
```

Output

```
Java Programming
```

Note

- *Both are treated as different variables.*

Rule 7: It is recommended to use meaningful and readable variable names.

Program

```
Subject_Name = "Python Programming"
print(Subject_Name)
```

Output

```
Java Programming
```

Deleting a Variable

Keyword "del" is used to delete a variable from memory. After deletion, the variable can no longer be accessed.

Program

```
i = 10
del i
print("Value of i=",i)
```

Output

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 3
      1 i = 10
      2 del i
----> 3 print("Value of i=",i)

NameError: name 'i' is not defined
```

Fix Code

Explanation

In this program:

- `i = 10`
  A variable `i` is created and assigned the value 10.
- `del i`
  The del statement deletes the variable `i` from memory.
- `print("Value of i=", i)`
  After deletion, the program tries to print the value of `i`.

Since the variable no longer exists, Python generates an error.

This happens because the variable i was deleted before it was used in the `print( )` statement.

Different ways to assign values to variables

1. Basic assignment

Basic assignment in Python means assigning a value to a variable using the = operator.

Program

```
a = 5
b = 3.2
c = "Python Programming"
print("value of a: ",a)
print("value of b: ",b)
print("value of c: ",c)
```

Output

```
value of a:  5
value of b:  3.2
value of c:  Python Programming
```

2. Dynamic Typing

Python is dynamically typed, which means a variable can hold different types of data at different times during program execution.

Program

```
var1=5
var1="Python Programming"
print(var1)
```

Output

```
Python Programming
```

3. Assigning Same Value

In Python, the same value can be assigned to multiple variables in a single line.

Program

```
a = b = c = 5
print("value of a : ",a)
print("value of b : ",b)
print("value of c : ",c)
```

Output

```
Python Programming
```

4. Assigning Different Values

Multiple variables can be also assigned different values in a single line.

Program

```
a, b, c = 5, 3.2, "Python Programming"
print("value of a: ",a)
print("value of b: ",b)
print("value of c: ",c)
```

Output

```
value of a:  5
value of b:  3.2
value of c:  Python Programming
```

1.4 Python Data Types

Data types in Python specify the type of value a variable can store. A variable can store data of different types. Data types define the characteristics of data, type of values a variable can store, the operations that can be performed on it, and the amount of memory it occupies.

Remember

- *Python automatically identifies the data type based on the assigned value.*

Built-in Data Types of Python

Python supports a wide variety of built-in Data Types as given below:

Table 1.1. Python Data Type

Data Type	Description	Example
`int`	Stores whole numbers	`i = 5`
`float`	Stores decimal numbers	`f = 2.3`
`str`	Stores text or characters	`S= "Python Programming"`
`bool`	Stores True or False values	`B = True`
`list`	Stores multiple items in a sequence	`L = [1, 2, 3]`
`tuple`	Stores ordered and unchangeable items	`T = (10, 20)`
`set`	Stores unordered unique items	`S = {1, 2, 3}`
`dict`	Stores data in key-value pairs	`D = {"Name":"Neha","Marks":19}`

Getting the Data Types

The data type of any variable can be identified using the `type( )` function:

Program

```
i = 5
f=2.3
c='M'
s="Python"
print("Type of variable i is : ",type(i))
print("Type of variable f is : ",type(f))
print("Type of variable c is : ",type(c))
print("Type of variable s is : ",type(s))
```

Output

```
Type of variable iis :<class 'int'>
Type of variable f is :<class 'float'>
Type of variable c is :<class 'str'>
Type of variable s is :<class 'str'>
```

(i) integers

An integer (int) is a numeric data type used to store numeric value i.e. whole numbers without decimal points. Integers can be positive, negative, or zero.

Program

```
x = 5
y = -5
z = 0
print("value of x : ",x)
print("value of y : ",y)
print("value of z : ",z)
```

Output

```
value of x :  5
value of y :  -5
value of z :  0
```

(ii) floats

A float (float) is a numeric data type used to store decimal or fractional value. A float value may be positive or negative.

Program

```
f1 = 5.0
f2 = -5.0
print("value of floating point variable f1 : ",f1)
```

```
print("value of floating point variable f2 : ",f2)
```

Output

```
value of floating point variable f1 :  5.0
value of floating point variable f2 :  -5.0
```

Here, 5.0 and -5.0 are floating point values.

(iii) strings

A string (str) is a data type used to store text or a sequence of characters enclosed within single quotes (' ') or double quotes (" ").

Remember

- *Strings in python are surrounded by either single quotation marks, or double quotation marks.*

'Python' is the same as "Python".

Program

```
S1 = "Python"
S2 = "Programming"
print("String S1 :",S1)
print("String S2 :",S2)
```

Output

```
String S1 : Python
String S2 : Programming
```

Here,"Python"and 'Programming'are string values.

(iv) booleans

In programming, it is often necessary to determine whether an expression is `True` or `False`. A boolean(bool) data type stores only two values: True and/or False. When two values are compared,

Remember

- *In Python, any expression can be evaluated to produce one of these two Boolean values: True or False.*
- *Boolean data type is commonly used in decision-making statements where some conditions are given.*

Program

```
age = 20
result = age >= 18
print("Is the person eligible to vote?", result)
```

Output

```
Is the person eligible to vote? True
```

Explanation

In this program, `age = 20` stores the age of a `person.age >= 18` checks whether the age is greater than or equal to `18`.

The expression returns a Boolean value:

`True if the condition is satisfied.`

`False if the condition is not satisfied.`

Since `20` is greater than `18`, the output is True.

1.5 Type Casting

Type casting is the process of converting a data type of a variable into another data type. It is also known as type conversion.

Python provIDE built-in functions to perform type conversion.

Types of Type Casting

Type casting can be broadly categorized into two types:

1. Implicit Type Casting (Automatic Conversion)
2. Explicit Type Casting (Manual Conversion)

1. Implicit Type Casting (Automatic Conversion)

Implicit Type Casting is the Automatic Conversion of one data type into another compatible data type by Python. It happens without any instruction from the programmer, usually when an operation involves different data types.

Remember

- *Python generally converts a smaller data type to a larger one to preserve the accuracy of the result and avoid data loss.*

Example

When an integer and a float are used in the same expression, Python automatically converts the integer into a float before performing the calculation.

Conversion of other Data Types to integer (int)

Converts values of other data type into an integer by removing the decimal part (for floats) or converting. Compatible strings and Boolean values.

Program

```
a = int(5)
b = int(3.2) # conversion from float to int
```

```
c = int("3") # conversion from string to int
print("Addition of a, b and c:",a+b+c)
```

Output

```
Addition of a, b and c: 11
```

Conversion of other Data Types to float (Float)

Converts values of other data type into a floating paint with a decimal point.

Program

```
a = float(5)   # conversion from int to float
b = float(3.2)
c = float("3") # conversion from string to float
print("Addition of a, b and c:",a+b+c)
```

Output

```
Addition of a, b and c: 11.2
```

Conversion of other Data Types to String (str)

Converts values of any data type into their string (teat) representation.

Program

```
a = str('S')
b = str(5) # conversion from float to float
c = str(1.2) # conversion from string to float
print("Addition of a, b and c:",a+b+c)
```

Output

```
Addition of a, b and c: S51.2
```

Benefit of Implicit Type Casting

Implicit Type Casting makes programming easier and reduces the need for manual type conversion.

2. Explicit Type Casting (Manual Conversion)

Explicit Type Casting is the process of manually converting one data type into another using Python's built-in functions such as `int()`, `float()`, `str()`, and `bool()`. It requires the programmer to explicitly specify the type conversion according to the needs of the program.

Remember

- *Explicit type casting is often used when converting a larger data type to a smaller one, such as converting a float to an integer, or when converting between different data types, such as a string to an integer.*

Example

When a number is stored as a string and needs to be used for mathematical calculations, the programmer must explicitly convert the string into an integer using the `int()` function.

Program

```
a = input("Enter a: ")
b = input("Enter b: ")
result = a*b
print("Multiplication a of and b :", result)
```

Output

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[25], line 4
      1 a = input("Enter a: ")
      2 b = input("Enter b: ")
----> 4 result = a*b
      6 print("Multiplication of a and b :", result)

TypeError: can't multiply sequence by non-int of type 'str'
```

Program

```
a = input("Enter a: ")
b = input("Enter b: ")
#Type casting the input strings to integers
a = int(a)
b = int(b)
result = a*b
print("Multiplication of a and b:", result)
```

Output

```
Enter a:  2
Enter b:  3
Multiplication of a and b: 6
```

1.6 Basic Operators

Operators are special symbols that are used to perform operations on variables, values, or expressions. They help manipulate data and produce a desired result. The values or variables on which operators act are called operands.

For example, the addition operator (+) is used to add two numbers or expressions together.

Program

```
a = 2
b = 3
result = a + b
print("Addition of a and b: ",result)
```

Output

```
Addition of a and b: 5
```

Explanation

In this program, operator is,+ (addition operator) and operands are `a` and `b` (10 and 5).

Remember

- ***Operators :*** *Operators are special symbols such as +, -, *, /, %, etc., that perform specific operations.*
- ***Operands :*** *Operands are the values, variables, or expressions on which an operator performs an operation.*

Types of Operators in Python

Python supports a variety of operators, including arithmetic, assignment, comparison, logical, bitwise, membership, and identity operators.

S. No.	Operator Type	Description	Example
1.	Arithmetic Operators	Perform mathematical calculations.	+, –, *, /, %, **, //
2.	Assignment Operators	Assign values to variables.	=, +=, -=, *=, /=, %=
3.	Comparison (Relational) Operators	Compare two values and return True or False.	==, !=, >, <, >=, <=
4.	Logical Operators	Combine conditional statements.	AND, OR, NOT
5.	Bitwise Operators	Perform operations on binary representations of integers.	&, `
6.	Membership Operators	Test whether a value exists in a sequence.	in, not in
7.	Identity Operators	Compare the memory locations of two objects.	is, is not

Arithmetic Operators

Arithmetic Operators are used to perform basic mathematical operations like addition, subtraction, multiplication and division.

Program

```
a = 5
b = 3
print("For variable a and b :")
print("Addition: ",a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
print("Modulus:", a % b)
```

Output

```
For variable a and b :
Addition:  8
Subtraction: 2
Multiplication: 15
Division: 1.6666666666666667
Modulus: 2
```

Note

- *In Python, the division operator (/) returns a floating-point result, while floor division (//) returns an integer result. Observe the program given below :*

Program

```
a = 5
b = 3
print("Division:", a / b)
print("Floor Division:", a // b)
```

Output

```
Division: 1.6666666666666667
Floor Division: 1
```

Comparison Operators

Comparison operators compares two values or expressions. It either returns True or False according to the condition. It is also called as Relational Operators.

Program

```
a = 2
b = 3
print("a > b: ",a > b)
print("a < b: ",a < b)
print("a == b: ",a == b)
print("a != b: ",a != b)
print("a >= b: ",a >= b)
print("a <= b: ",a <= b)
```

Output

```
a > b:  False
a < b:  True
a == b:  False
a != b:  True
a >= b:  False
a <= b:  True
```

Logical Operators

Logical operators are used to combine or modify conditional statements/expressions. They return a Boolean Value (True or False) based on the evaluation of the conditions.

Logical and

"Logical and" returns True if both statements are true.

Program

```
print(False and False)
print(False and True)
print(True and False)
print(True and True)
```

Output

```
False
False
False
True
```

Logical or

"Logical or" returns True if one of the statements is true.

Program

```
print(False or False)
print(False or True)
print(True or False)
print(True or True)
```

Output

```
False
True
True
True
```

Logical not

"Logical not" reverse the result, returns False if the result is true.

Program

```
print(not True)
print(not False)
```

Output

```
False
True
```

Note

- *The precedence of Logical Operators in Python is as follows:*
  1. Logical not
  2. logical and
  3. logical or

1.7 Input and Output Operations

Every Computer Programming Language requires data to process and execute instructions. The data provided to a computer for processing is called input. After the data is processed, the result obtained is known as output.

In Computer Programming, input/output (I/O) refers to the communication between a computer program and the outside world. It involves receiving data from users, files, or external devices as input, and displaying or sending the processed information back as an output.

In Python Programming, input is any data that flows into your program that can be one of the following:

1. User input from the keyboard
2. Data read from files
3. Network requests

4. Signals from hardware sensors

Whereas, output is any form of data that flows out of your program to the outside world i.e.:

1. Text printed to the screen
2. Data written to files
3. Network responses
4. Signals sent to hardware

Python provIDE built-in functions like `input( )` for reading user input, `print()` for displaying output

Taking Input in Python

Input means taking data from the user. Python uses the `input( )` function to accept data from the user. This function pauses the execution of the program until the user enters input using the keyboard.

Program

```
Name = input("Enter your name: ")
print("Hello,", Name, "! Welcome to Python Programming")
```

Output

```
Enter your name:  Mayank
Hello, Mayank ! Welcome to Python Programming
```

Explanation

This program prompts the user to input their name, stores it in the variable `"name"` and then prints a greeting message addressing the user by their entered name.

Displaying Output in Python

Output means displaying information on the screen. Python uses the `print( )` function to display output on the screen.

`print( )` function allows a programmer to display text, variables and expressions on the console.

Program

```
print("Python Programming")
```

Output

```
Python Programming
```

In this program, `"Python Programming"` is a string literal which is enclosed within double quotes. When this line of code being executed, it outputs the text to the console.

Different types of output statements

1. Print with "end" Parameter

In Python, the `print()` function is used to display output on the screen. By default, after printing the given text or value, the `print ()` function moves the cursor to the next line. This happens because the default value of the `end` parameter is `"\n"` (newline character).

The `end` **parameter** allows the programmer to specify what should be printed at the end of the output instead of the default newline. It is useful when multiple outputs need to be displayed on the same line or separated by a specific character.

Program

```
print("Good Morning!", end=" "),
print("Welcome to Python Programming")
```

Output

```
Good Morning! Welcome to Python Programming
```

Explanation

By default, every `print( )` function moves the cursor to a new line after printing the output. In this program we have included the `end=" "` after the end of the first `print( )` statement.Therefore, you get the output in a single line separated by space.

2. Print with "sap" Parameter Concatenated Strings

When multiple values are printed together, Python automatically separates them with a space. The `"sep"` parameter is used to change this default separator.

Program

```
print("12", "December", "2013", sep="-")
```

Output

```
12-December-2013
```

Explanation

In this program, we have used the optional parameter sep= "-" inside the `print()` statement. As a result, the output includes items separated by - not comma.

Note

- *`"sep"` stands for separator. It specifies the character or symbol placed between multiple values.*

3. Print with Concatenated Strings

In Python, two or more strings can be joined together inside the `print( )` statement using the + operator. This process is called string concatenation. The +

operator joins the strings together and displays them as a single string.Refer the program given below:

Program

```
print('Python Programming'+ ' is easy to code')
```

Output

```
Python Programming is easy to code
```

1.8 Basic String Operations and Manipulation Techniques

What is String?

In Python, a string is a sequence of characters enclosed within single quotes (' ') or double quotes (" "). For example, "Python" is a string containing a sequence of characters 'P', 'y', 't', 'h', 'o', and 'n'.

Remember

- *Strings are used to store text data.*

Basic String Operations

Basic String Operations are the fundamental actions that can be performed on strings to create, access, combine, and extract information from text data.

Some of basic string operations are listed below:

1. Creating Strings
2. String Indexing
3. String Concatenation
4. String Repetition
5. String Slicing
6. Finding the length of String

1. Creating Strings

Strings can be created using single quotes, double quotes, or triple quotes.

(i) Creating String with Single Quotes

Program

```
s='Hello World'
print(s)
```

Output

```
Hello World
```

Explanation

In this program, we have created a string variable named 's'. The variable is initialized with the string 'Hello World'. We have used single quotes to represent strings, but we can use double quotes too.

(ii) Creating String with Double Quotes

Program

```
s="Welcome to Python Programming"
print(s)
```

Output

```
Welcome to Python Programming
```

Explanation

In this program, we have created a string variable named 's'. The variable is initialized with the string "Welcome to Python Programming". We have used double quotes to represent strings, but we can use triple quotes too.

(iii) Creating String with Triple Quotes

Multi-line strings in Python are created using triple single quotes ('''...''') or triple double quotes ("""..."""). They allow text to extend across multiple lines, and Python preserves the line breaks exactly as written. This feature is useful for storing long text, documentation, or formatted messages.

Program

```
s = """Python is an
Object Oriented Programming"""
print(s)
s = '''This program demonstrates
the use of Multi Line String'''
print(s)
```

Output

```
Python is an
Object Oriented Programming
This program demonstrates
the use of Multi Line String
```

2. String Indexing

Indexing is used to access individual characters in a string.

Program

```
s = "Python Programming "
print(s[0])
print(s[5])
print(s[7])
```

Output

```
print(s[0])
print(s[5])
print(s[7])
```

3. String Concatenation (Joining Strings)

Strings can be joined by using + operator in Python. The process of joining two or more strings is called as Concatenation.

Program

```
s1="Python "
s2="Programming"
s3=s1+s2
print(s3)
```

Output

```
Python Programming
```

4. String Repetition

The * operator can be used to repeat a string for multiple times.

Program

```
s = "Python Programming "
print(s*3)
```

Output

```
Python Programming Python Programming Python Programming
```

5. String Slicing

Slicing is used to extract a specific part of a string. It allows you to obtain a subset of characters from a string.

Program

```
s = "Python Programming "
print(s[0:4])
print(s[7:11])
```

Output

```
Pyth
Prog
```

6. Finding the length of String

The `len( )` function is used to find the number of characters in a string.

Program

```
s="Python Programming"
print("Lengh of String : ",len(s))
```

Output

```
Lengh of String :  18
```

String Manipulation Techniques

String manipulation techniques are the various methods and operations used to modify, access, combine, or process strings in Python. These techniques help programmers work with text data efficiently.

String manipulation includes tasks such as :

1. Changing Case (Uppercase and Lowercase letters)
2. Replacing Text (Words or Characters)
3. Splitting strings
4. Removing extra spaces
5. Extracting characters or portions of a string

These techniques make it easier to store, edit, and display text data in Python programs.

1. Changing Case/Modify String

Python provIDE functions to change the case of strings.

Program

```
s="PythonProgramming"
print(s.upper())
print(s.lower())
```

Output

```
PYTHON PROGRAMMING
python programming
```

2. Replacing Text (Word/Characters)

The `replace()` method is used to replace one word or character with another.

Program

```
s="Python Tutorial"
print(s.replace("Tutorial","Programming"))
```

Output

```
Python Programming
```

Program

```
S = "Hello! from Python"
print(S.replace("H", "C"))
```

Output

```
Cello! from Python
```

3. Splitting Strings

The `split( )` methoda string into parts.

Program

- `s="Python is an Object Oriented Programmingprint(s.split( ))`

Output

```
['Python', 'is', 'an', 'Object', 'Oriented', 'Programming']
```

4. Removing Extra Spaces

The `strip( )` method removes extra spaces from the beginning and end of a string.

Program

```
s="Python Programming"
print(s.strip())
```

Output

```
Python Programming
```

5. Extracting Characters or Portions of a String

You can extract characters or a part of a string using indexing.

(i) Extract from the Beginning

Program

```
s = "Python Programming "
print(s[:4])
```

Output

```
Pyth
```

(ii) Extract upto the End

Program

```
s = "Python Programming "
print(s[7:])
```

Output

```
Programming
```

Remember

- *Basic String Operations help you access and combine strings, whereas String Manipulation Techniques help you modify, format, and process string data to achieve specific tasks.*

1.9 Introduction to Standard Libraries

Python provIDE a large collection of built-in libraries called the **Python Standard Library.** These libraries come pre-installed with Python, so no separate installation is needed. They contain ready-made functions and modules that help programmers perform common tasks easily.

Some commonly used standard libraries are:

- **math** - used for mathematical calculations
- **os** - used to interact with the operating system
- **datetime** - used for date and time operations
- **random** - used to generate random numbers
- **json** - used to work with JSON data

The standard library makes programming easier, faster, and more efficient.

Third-party libraries

Third-party libraries are libraries developed by other programmers or organizations. These libraries are not included with Python by default and must be installed separately using the pip command.

Third-party libraries provide advanced features for different fields such as data science, machine learning, web development, and game development.

Some popular third-party libraries are:

- **NumPy** - used for numerical and scientific calculations
- **Pandas** - used for data analysis and handling tables of data
- **Matplotlib** - used for creating graphs and charts
- **TensorFlow** - used for machine learning and deep learning
- **PyGame** - used for developing games

Third-party libraries help developers build powerful applications with less coding effort.

Installing and Uninstalling libraries

Python libraries can be added or removed using the pip package manager. Pip helps users install, update, and uninstall Python libraries easily.

Installing a Library

You can use the following command in the command prompt or terminal to install a library in Python:

1. `pip install numpy`
2. `pip install pandas`
3. `pip install scipy`

4. `pip install matplotlib`
5. `pip install scikit-learn`

Uninstalling a Library

You can use the following command in the command prompt or terminal to remove a library from Python:

1. `pip uninstall numpy`
2. `pip uninstall pandas`
3. `pip uninstall scipy`
4. `pip uninstall matplotlib`
5. `pip uninstall scikit-learn`

Execute these commands in the Windows Command Prompt. We will discuss these commands in details, in the upcoming units.

Remember

- ***PIP - Preferred Installer Program***
- *PIP is the package manager used in Python Programming to install, update, and remove Python libraries and packages.*

EXERCISE

Multiple Choice Questions (MCQs)

1. Which of the following is an Integrated Development Environment (IDE) for Python?

(a) Notepad (b) MS Paint
(c) PyCharm (d) Calculator

Answer: (c) PyCharm

2. Which of the following is NOT an IDE?

(a) Jupyter Notebook (b) VS Code
(c) PyCharm (d) Notepad

Answer: (d) Notepad

3. What is one major advantage of an IDE over a text editor?

(a) Automatic debugging tools
(b) Syntax highlighting
(c) Code completion
(d) All of the above

Answer: (d) All of the above

4. **Which symbol is used for variable assignment in Python?**
   (a) == (b) =
   (c) := (d) !=
   Answer: (b) =
5. **Which of the following is a valid variable name?**
   (a) 2num (b) class
   (c) student_name (d) first-name
   Answer: (c) student_name
6. **Which data type stores whole numbers?**
   (a) float (b) bool
   (c) int (d) str
   Answer: (c) int
7. **Which data type stores decimal values?**
   (a) int (b) float
   (c) bool (d) str
   Answer: (b) float
8. **Which of the following is a string?**
   (a) 123 (b) True
   - (c) "Python(d) 10.5

   Answer: (c) "Python"
9. **What is the output of the following code?**
   print(type(True))
   (a) int (b) str
   (c) bool (d) float
   Answer: (c) bool
10. **What is type casting?**
    (a) Creating variables
    (b) Converting one data type into another
    (c) Deleting variables
    (d) Comparing variables
    Answer: (b) Converting one data type into another
11. **Which function converts a string into an integer?**
    (a) float() (b) bool()
    (c) str() (d) int()
    Answer: (d) int()
12. **Which operator is used for addition?**

(a) * (b) +
(c) / (d) %

Answer: (b) +

13. Which operator is used to compare equality?

(a) = (b) ==
(c) != (d) >=

Answer: (b) ==

14. Which of the following is a logical operator?

(a) > (b) +
(c) and (d) %

Answer: (c) and

15. Which function is used to accept input from the user?

(a) `print()` (b) `input()`
(c) read() (d) get()

Answer: (b) `input()`

16. Which function is used to display output?

(a) display() (b) output()
(c) `print()` (d) show()

Answer: (c) `print()`

17. Which operator is used for string concatenation?

(a) * (b) +
(c) / (d) %

Answer: (b) +

18. Which method converts a string to uppercase?

(a) lower() (b) capitalize()
(c) upper() (d) title()

Answer: (c) upper()

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

(a) pip remove (b) pip install
(c) pip library (d) install pip

Answer: (b) pip install

20. Which command is used to uninstall a Python library?

(a) pip remove
(b) remove pip
(c) pip uninstall
(d) uninstall library

Answer: (c) pip uninstall

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

(a) math (b) random

(c) pandas (d) os

Answer: (c) pandas

22. What will be the output?

```
print(bool(0))
```

(a) True (b) False

(c) 0 (d) Error

Answer: (b) False

23. What will be the output?

```
x = 10
y = 5
print(x + y)
```

(a) 15 (b) 105

(c) 50 (d) Error

Answer: (a) 15

24. Which operator is used for exponentiation in Python?

(a) ^ (b) **

(c) * (d) //

Answer: (b) **

25. Which library is commonly used for data analysis in Python?

(a) Pandas (b) Matplotlib

(c) SciPy (d) NumPy

Answer: (a) Pandas

▼ Short Answer Questions

1. What is Python?
2. What is an Integrated Development Environment (IDE)?
3. Name any three IDE used for Python programming.
4. Differentiate between a text editor and an IDE.
5. List the steps involved in installing Python.
6. What is a variable? Give an example.
7. State the rules for naming variables in Python.
8. What is dynamic typing in Python?
9. What are data types?

10. Explain the integer data type with an example.
11. Explain the float data type with an example.
12. Explain the string data type with an example.
13. Explain the Boolean data type with an example.
14. Define type casting.
15. What is implicit type casting?
16. What is explicit type casting?
17. Write statements to convert :
    (i) Integer to float
    (ii) Float to integer
    (iii) String to integer
18. What are arithmetic operators?
19. What are comparison operators?
20. What are logical operators?
21. Differentiate between comparison operators and logical operators.
22. What is the purpose of the `input()` function?
23. What is the purpose of the `print()` function?
24. Write a Python program to input a student's name and display it.
25. What is a string?
26. What is string concatenation?
27. What is string repetition?
28. Explain string indexing.
29. Explain string slicing.
30. Name any four string methods.
31. What is a standard library?
32. What is a third-party library?
33. Give examples of standard libraries.
34. Give examples of third-party libraries.
35. Write the command to install NumPy.
36. Write the command to uninstall Pandas.

▼ Long Answer Questions

1. Explain the installation process of Python and Jupyter Notebook.
2. Explain the installation process of PyCharm and VS Code.
3. Discuss the advantages of IDE over text editors.

4. Explain Python variables and their characteristics with examples.
5. Discuss variable naming conventions in Python.
6. Explain various Python data types with suitable examples.
7. Differentiate between integers, floats, strings, and Boolean values.
8. Explain implicit and explicit type casting with examples.
9. Discuss the advantages of type casting in Python.
10. Explain arithmetic operators with suitable examples.
11. Explain comparison operators with suitable examples.
12. Explain logical operators with suitable examples.
13. Write a Python program to perform arithmetic and comparison operations on two numbers.
14. Explain input and output operations in Python with examples.
15. Write a Python program to accept two numbers and display their sum, difference, product, and quotient.
16. Explain string operations in Python with examples.
17. Discuss various string manipulation techniques with examples.
18. Write a Python program to perform different string operations on a user-entered string.
19. Differentiate between standard libraries and third-party libraries.
20. Explain the procedure for installing and uninstalling Python libraries using pip.
21. Discuss the importance of third-party libraries such as NumPy, Pandas, SciPy, Matplotlib,and Scikit-learn in Python programming.

▼ Programming Exercise

1. Write a Python program to manage the inventory of a pharmacy. The program should allow the user to add, remove, update, and display medication records. It should also calculate the total value of medicines available in stock.
2. Write a Python program to calculate the dosage of a medication based on the patient's weight and the recommended dosage per kilogram of body weight. The program should display a warning message if the calculated dosage exceeds a specified limit.
3. Write a Python program to maintain patient records in a clinic. The program should allow the user to enter patient details such as name, age, weight, and diagnosis, and display the stored information.

4. Write a Python program to calculate the Body Mass Index (BMI) of patients. The program should accept height and weight as input, calculate BMI, and classify the patient as underweight, normal, overweight, or obese.
5. Write a Python program to calculate the total cost of medicines purchased by a customer. The program should accept medicine names, quantities, and prices and generate a bill.
6. Write a Python program to manage a hospital pharmacy inventory. The program should keep track of medicine names, quantities available, and expiry dates, and identify medicines that are low in stock.
7. Write a Python program to calculate a patient's daily fluid requirement based on body weight. The program should display the total fluid requirement and indicate whether the patient requires special monitoring.
8. Write a Python program to store and analyze patients' blood pressure readings. The program should calculate the average reading and identify whether the patient has normal, elevated, or high blood pressure.
9. Write a Python program to manage vaccination records. The program should allow the user to add, update, search, and display vaccination details of patients.
10. Write a Python program to calculate the percentage of active pharmaceutical ingredient (API) present in a drug sample. The program should accept sample weight and API weight as input and display the percentage purity.
11. Write a Python program to manage laboratory test reports. The program should store patient details and test results and display a summary report.
12. Write a Python program to calculate the concentration of a drug solution. The program should accept the amount of drug and volume of solvent as input and display the concentration.
13. Write a Python program to analyze medicine sales data. The program should calculate the total sales, average sales, and the medicine with the highest sales.
14. Write a Python program to maintain a database of pharmaceutical products. The program should allow the user to add product details, search for a product, and display all available products.
15. Write a Python program to calculate the infusion rate for intravenous (IV) fluids. The program should accept the volume to be infused and infusion time and calculate the flow rate.

16. Write a Python program to analyze patient temperature records for a week. The program should calculate the average temperature and identify abnormal readings.
17. Write a Python program to manage prescription information. The program should store patient details, prescribed medicines, dosage schedules, and display prescription summaries.
18. Write a Python program to calculate the age of a medicine batch from its manufacturing date and determine whether it is nearing expiry.
19. Write a Python program to perform various string operations on medicine names, such as converting to uppercase, counting characters, searching for a specific word, and extracting substrings.
20. Write a Python program that uses the NumPy, Pandas, and Matplotlib libraries to analyze pharmaceutical sales data and display the results graphically.