Program	B.Pharmacy
Semester	First
CourseTitle /Subject Name	Basics of Python Programming for Pharmaceutical Sciences (Theory)
Course Code	BP101T
Module	Data Structures & File Handling
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.

Learning Outcomes:

After completing this unit, students will be able to:

LO	**Learning Outcomes**	CO
1	Understand Python Data Structures	BP101.3
2	Perform Operations on Lists and Dictionaries	BP101.3
3	Apply Indexing and Slicing Techniques	BP101.3
4	Perform String Manipulation	BP101.3
5	Understand and Use NumPy Arrays	BP101.3
6	Handle CSV Files	BP101.3
7	Understand Structured Healthcare Datasets	BP101.3
8	Import and Analyze Pharmaceutical Datasets	BP101.3
9	Apply Data Handling Skills to Healthcare Problems	BP101.3
10	Develop Practical Data Analysis Skills	BP101.3

Data Structures & File Handling

Introduction

A collection is a single object representing a group of objects (such as a list or dictionary). Collections may also be referred to as containers (as they contain other objects). These collection classes are often used as the basis for more complex or application specific data structures and data types. These collection types support various types of data structures (such as lists and maps) and ways to process elements within those structures. This chapter introduces the Python Collection types.

Python Collection Types

There are four classes in Python that provide container like behaviour; that is data types for holding collections of other objects, these are:

• **Tuples:**

A tuple is an ordered collection of elements that can store different types of data. Tuples are immutable, which means their values cannot be changed after they are created.

Syntax

```
</>python
my_tuple = (10, 20, 30)
```

Example:

```
</>python
student = ("Ram", 21, "B.Pharm")
print(student)
```

Output:

```
</>python
('Ram', 21, 'B.Pharm')
```

• **Lists**:

A **list** is an ordered and mutable collection of items in Python. It can store different types of data such as numbers, strings, and boolean values. Lists are created using square brackets `[]`.

Syntax:

```
</>python
list_name = [item1, item2, item3]
```

Example:

```
</>python
fruits = ["Apple", "Banana", "Mango"]
print (fruits)
```

Output:

```
</>python
['Apple', 'Banana', 'Mango']
```

• **Dictionaries**:

A **dictionary** is a collection of data stored in **key-value pairs**. Dictionaries are **ordered, mutable**, and do not allow duplicate keys. They are created using curly braces `{}`.

Syntax:

```
</>python
dictionary_name = {
    key1: value1,
    key2: value2
}
```

Example:

```
</> python
student = {
    "name": "Sid",
    "age": 21,
    "course": "B.Pharm"
}

print(student)
```

Output:

```
</>python
{'name': 'Raman', 'age': 21, 'course': 'B.Pharm'}
```

Indexing and Slicing Lists in Python

1. Indexing

Indexing is used to access individual elements of a list using their position (index number).

Example:

```
</>python
fruits = ["Apple", "Banana", "Mango", "Orange"]
print(fruits[0])   # First element
print(fruits[2])   # Third element
print(fruits[-1])  # Last element
```

Output:

```
</>python
Apple
Mango
Orange
```

Types of Indexing

- **Positive Indexing:** Starts from 0
  - Apple → 0
  - Banana → 1
  - Mango → 2
  - Orange → 3
- **Negative Indexing:** Starts from -1
  - Orange → -1
  - Mango → -2
  - Banana → -3
  - Apple → -4

2. Slicing

Slicing is used to extract a portion (sublist) of a list.

Syntax:

```
</>python
list_name[start : stop : step]
```

- **start** → Starting index (included)
- **stop** → Ending index (excluded)
- **step** → Interval between elements

Example:

```
</>python
fruits = ["Apple", "Banana", "Mango", "Orange", "Grapes"]
print (fruits[1:4])
print (fruits[:3])
print (fruits[2:])
print (fruits[::2])
```

```
Output:
</> python
['Banana', 'Mango', 'Orange']
['Apple', 'Banana', 'Mango']
['Mango', 'Orange', 'Grapes']
['Apple', 'Mango', 'Grapes']
```

Basic Operations on Lists and Dictionaries in Python

1. Basic Operations on Lists

A list is an ordered and mutable collection of elements.

Creating a List

```
</>python
fruits = ["Apple", "Banana", "Mango"]
```

Accessing Elements

```
</>python
print(fruits[0])
```

Output:

```
</>python
Apple
```

Adding Elements

```
</>python
fruits.append("Orange")
print(fruits)
```

Output:

```
</>python
['Apple', 'Banana', 'Mango', 'Orange']
```

Removing Elements

```
</>python
fruits.remove("Banana")
print(fruits)
```

Output:

```
</>python
['Apple', 'Mango', 'Orange']
```

Updating Elements

```
</>python
fruits[1] = "Grapes"
print(fruits)
```

Output:

```
</>python
['Apple', 'Grapes', 'Orange']
```

Finding Length

```
</>python
print(len(fruits))
```

Output:

```
</>python
3
```

2. Basic Operations on Dictionaries

A **dictionary** stores data in key-value pairs.

Creating a Dictionary

```
</>python
student = {
    "name": "Rohit",
    "age": 21,
    "course": "B.Pharm"
}
```

Accessing Values

```
</>python
print(student["name"])
```

Output:

```
</>python
Rohit
```

Adding a New Key-Value Pair

```
</>python
student["city"] = "Delhi"
print(student)
```

Updating a Value

```
</>python
student["age"] = 22
print(student)
```

Output:

```
</>python
{'name': 'Rohit', 'age': 22, 'course': 'B.Pharm', 'city': 'Delhi'}
```

Removing a Key-Value Pair

```
</>python
del student["city"]
print(student)
```

Output:

```
</>python
{'name': 'Rohit', 'age': 22, 'course': 'B.Pharm'}
```

Finding Length

```
</>python
print(len(student))
```

Output:

```
</>python
3
```

String manipulation techniques

String manipulation techniques are methods and operations used to create, modify, search, split, join, and format strings in Python.

1. Concatenation (Joining Strings)

Combines two or more strings using the + operator.

Example

```
</>python
first_name = "Sahil"
last_name = "Kumar"
full_name = first_name + " " + last_name
print(full_name)
```

Output:

```
</>python
Sahil Kumar
```

2. Repetition

Repeats a string using the * operator.

Example

```
</>python
text = "Hello "
print(text * 3)
```

Output:

```
</>python
Hello Hello Hello
```

3. Changing Case

Example

```
</>python
text = "Python Programming"
print(text.upper())
print(text.lower())
print(text.title())
```

Output:

```
</>python
PYTHON PROGRAMMING
python programming
Python Programming
```

4. Replacing Characters or Words

Example

```
</>python
text = "I like Java"
new_text = text.replace("Java", "Python")
print(new_text)
```

Output:

```
</>python
I like Python
```

5. Finding a Substring

Example

```
</>python
text = "Python Programming"
print(text.find("Program"))
```

Output:

```
</>python
7
```

6. Splitting a String

Converts a string into a list.

Example

```
</>python
text = "Apple,Banana,Mango"
fruits = text.split(",")
print(fruits)
```

Output:

```
</>python
['Apple', 'Banana', 'Mango']
```

---

7. Joining Strings

Joins list elements into a single string.

Example

```
</>python
fruits = ["Apple", "Banana", "Mango"]
result = ", ".join(fruits)
print(result)
```

Output:

```
</>python
Apple, Banana, Mango
```

8. Removing Extra Spaces

Example

```
</>python
text = "   Python   "
print(text.strip())
```

Output:

```
</>python
Python
```

9. Checking String Content

Example

```
</>python
text = "Python123"
print(text.isalpha())
print(text.isdigit())
print(text.isalnum())
```

Output:

```
</>python
False
False
True
```

10. String Length

Example

```
</>python
text = "Python"
print(len(text))
```

Output:

```
</>python
6
```

Introduction to NumPy arrays

A **NumPy array** is a collection of elements of the same data type stored in a single variable. NumPy arrays are represented by the ndarray object.

Advantages of NumPy Arrays

- Faster than Python lists.
- Requires less memory.
- Supports mathematical operations.
- Useful for scientific and data analysis applications.
- Can handle large datasets efficiently.

1. Array Creation

Creating a One-Dimensional Array

```
</>python
import numpy as np
arr = np.array([10, 20, 30, 40, 50])
print(arr)
```

Output:

```
</>python
[10 20 30 40 50]
```

Creating a Two-Dimensional Array

```
</>python
import numpy as np
arr = np.array([[1, 2, 3],
                [4, 5, 6]])
print(arr)
```

Output:

```
</>python
[[1 2 3] [4 5 6]]
```

Creating Arrays with Special Functions

```
</>python
import numpy as np
print(np.zeros(5))
print(np.ones(5))
print(np.arange(1, 6))
```

Output:

```
</>python

[0. 0. 0. 0. 0.]
[1. 1. 1. 1. 1.]
[1 2 3 4 5]
```

2. Arithmetic Operations on Arrays

NumPy performs operations element-wise.

Addition

```
</>python
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print(a + b)
```

Output:

```
</>python
   [5 7 9]
```

Subtraction

```
</>python
print(b - a)
```

Output:

```
</>python
      [3  3  3]
```

Multiplication

```
 </>python
print(a * b)
```

Output:

```
</>python
[ 4 10 18]
```

Division

```
</>python
print(b / a)
```

Output:

```
</>python
 [4.  2.5 2. ]
```

3. Operations with a Scalar

Example

```
</>python
import numpy as np
arr = np.array([10, 20, 30])
print(arr + 5)
print(arr * 2)
```

Output:

```
</> python
[15 25 35]
[20 40 60]
```

4. Finding Sum, Mean, and Maximum

```
</>python
import numpy as np
arr = np.array([10, 20, 30, 40])
print(np.sum(arr))
print(np.mean(arr))
print(np.max(arr))
```

Output:

```
</>python
100
25.0
40
```

Reading and writing CSV files

Reading CSV Files:

Reading a CSV file in Python means retrieving data stored in a Comma-Separated Values (CSV) file and loading it into a Python program for processing, analysis, or display. This is commonly done using the built-in `csv` module or the `pandas` library. Reading CSV Loading data from a CSV file into a Python program.

Example CSV file (data.csv)

```
</>csv
Name,Age,City
John,25,New York
Alice,30,London
Bob,22,Paris
```

Reading a CSV File Using csv.reader

```
</>python
import csv
with open('data.csv', 'r') as file:
    csv_reader = csv.reader(file)
    for row in csv_reader:
        print(row)
```

Output

```
</>python
['Name', 'Age', 'City']
['John', '25', 'New York']
['Alice', '30', 'London']
['Bob', '22', 'Paris']
```

Writing CSV Files

Writing a CSV file in Python means creating a new CSV file or adding data to an existing CSV file by storing information in a tabular format, where values are separated by commas. This is usually done using the csv. Writer or csv.DictWriter class. Writing CSV Saving data from a Python program into a CSV file.

Writing Using csv.writer

```
</>python
import csv
data = [
    ['Name', 'Age', 'City'],
    ['John', 25, 'New York'],
    ['Alice', 30, 'London'],
    ['Bob', 22, 'Paris']
]
with open('output.csv', 'w', newline='') as file:
 csv_writer = csv.writer(file)
 csv_writer.writerows(data)
print("CSV file written successfully.")
```

Output:

```
</>python
CSV file written successfully
```

Understanding Structured Healthcare Datasets

A structured healthcare dataset is a collection of healthcare-related data organized in a well-defined format, such as rows and columns in a table. These datasets contain information about

patients, diseases, medications, laboratory test results, hospital visits, and treatment outcomes. Python is widely used to store, analyze, and visualize structured healthcare data.

Characteristics of Structured Healthcare Datasets

- Data is organized in rows and columns.
- Each row represents a record (e.g., a patient).
- Each column represents an attribute or variable (e.g., age, gender, diagnosis).
- Easy to store in CSV files, Excel sheets, or databases.

- Suitable for data analysis and machine learning.

Example of a Healthcare Dataset

Patient_ID	Name	Age	Gender	Disease	Blood_Pressure
101	John	45	Male	Diabetes	130/85
102	Alice	38	Female	Hypertension	145/90
103	Bob	50	Male	Asthma	120/80

Python Example Using Pandas

Creating a Healthcare Dataset

```
</>python
import pandas as pd

data = {
   "Patient_ID": ["P001", "P002", "P003", "P004"],
   "Patient_Name": ["Amit Kumar", "Neha Sharma", "Raj Singh", "Priya Gupta"],
   "Age": [45, 32, 60, 28],
   "Disease": ["Diabetes", "Fever", "Hypertension", "Allergy"],
   "Medicine": ["Metformin", "Paracetamol", "Amlodipine", "Cetirizine"]
}
df = pd.DataFrame(data)
print(df)
```

OUTPUT

	Patient_ID	Patient_Name	Age	Disease	Medicine
0	P001	Amit Kumar	45	Diabetes	Metformin
1	P002	Neha Sharma	32	Fever	Paracetamol
2	P003	Raj Singh	60	Hypertension	Amlodipine
3	P004	Priya Gupta	28	Allergy	Cetirizine

Common Operations on Healthcare Datasets

1. **Loading data** from CSV or Excel files.
2. **Viewing records** using head() and tail().
3. **Filtering patients** based on age, disease, or gender.
4. **Calculating statistics** such as average age or blood pressure.
5. **Handling missing values** in patient records.
6. **Visualizing data** using charts and graphs.

Importing Small Pharmaceutical Datasets and Performing Basic Data Access and Manipulation Tasks

Definition

Importing pharmaceutical datasets in Python refers to loading drug, patient, prescription, or clinical trial data from files such as CSV or Excel into Python for analysis. **Data access and** manipulation involve viewing, filtering, modifying, and analyzing the dataset using Python libraries such as `pandas`.

A small pharmaceutical dataset may contain information such as:

- Drug names
- Batch numbers
- Manufacturing dates
- Expiry dates
- Prices
- Stock quantities

Sample Pharmaceutical Dataset (CSV File)

pharma_data.csv

```
Drug_Name,Category,Price,Stock
Paracetamol,Analgesic,25,150
Amoxicillin,Antibiotic,80,75
Cetirizine,Antihistamine,40,120
Metformin,Antidiabetic,60,90
```

Method 1: Using Pandas Library

Step 1: Import Pandas

```
</>python
import pandas as pd
```

Step 2: Load the Dataset

```
</>python
data = pd.read_csv("pharma_data.csv")
print(data)
```

Output

```
    Drug Name      Category      Price      Stock
0   Paracetamol    Analgesic      25         150
1   Amoxicillin    Antibiotic     80         75
2    Cetirizine    Antihistamine  40         120
3    Metformin     Antidiabetic   60         90
```

Step 3: Display First Few Records

```
</>python
print(data.head())
```

Step 4: Display Dataset Information

```
</>python
print(data.info())
```

Output:

```
</>python
<class 'pandas.core.frame.DataFrame'>
4 entries
4 columns
```

Basic Data Access Operations

Access a Column

```
</>python
print(data["Drug_Name"])
```

Output:

```
</>python
0    Paracetamol
1    Amoxicillin
2     Cetirizine
3      Metformin
```

Access Multiple Columns

```
</>python
print(data[["Drug_Name", "Price"]])
```

Output:

```
</>python
    Drug_Name    Price
0   Paracetamol     25
1   Amoxicillin     80
2   Cetirizine      40
3   Metformin       60
```

Access a Specific Row

```
</>python
print(data.iloc[1])
```

Output:

```
</>python
Drug_Name    Amoxicillin
Category     Antibiotic
Price              80
Stock               75
```

Basic Data Manipulation Tasks

1. Add a New Column

Suppose GST is 5%.

```
</>python
data ["Price_With_GST"] = data["Price"] * 1.05
print (data)
```

2. Filter Data

Display medicines with stock greater than 100.

```
</>python
high_stock = data[data["Stock"] > 100]
print(high_stock)
```

Output:

```
</>python
    Drug_Name        Category      Price    Stock
0  Paracetamol       Analgesic       25      150
2   Cetirizine       Antihistamine   40      120
```

3. Update Values

Increase the price of Paracetamol by ₹5.

```
</>python
data.loc[data["Drug_Name"] == "Paracetamol", "Price"] += 5
print(data)
```

4.Calculate Average Price

```
</>python
avg_price = data["Price"].mean()
print("Average Price:", avg_price)
```

Output:

```
</>python
Average Price: 51.25
```

5. Sort Data

Sort medicines by price.

```
</>python
sorted_data = data.sort_values(by="Price")
print(sorted_data)
```

6. Remove a Column

```
</>python
data.drop("Stock", axis=1, inplace=True)
print(data)
```

Save Modified Dataset

```
</>python
data.to_csv("updated_pharma_data.csv", index=False)
print("Dataset saved successfully!")
```

Multiple Choice Questions (MCQs)

1. Which data structure is enclosed in square brackets [ ]?

A) Tuple
B) Dictionary
C) List
D) Set

Answer: C) List

2. Which data structure is immutable?

A) List
B) Dictionary
C) Tuple
D) Set

Answer: C) Tuple

3. Which symbol is used to create a dictionary?

A) ( )
B) [ ]
C) { }
D) < >

Answer: C) { }

4. What is the index of the first element in a Python list?

A) 1
B) -1
C) 0
D) 2

Answer: C) 05. What is the output of len([1,2,3,4])?

A) 3
B) 4
C) 5
D) Error

Answer: B) 4

6. Which method adds an item to a list?

A) add()
B) append()
C) insertItem()
D) join()

Answer: B) append()

7. Which method removes an item from a list?

A) remove()
B) delete()
C) erase()
D) clearItem()

Answer: A) remove()

8. Which function returns the largest value in a list?

A) high()
B) max()
C) large()
D) top()

Answer: B) max()

9. What is the output of [1,2,3][1]?

A) 1
B) 2
C) 3
D) Error

Answer: B) 2

10. What is list slicing?

A) Adding elements
B) Accessing part of a list
C) Sorting a list
D) Deleting a list

Answer: B) Accessing part of a list

11. What is the output of [10,20,30,40][1:3]?

A) [10,20]
B) [20,30]
C) [30,40]
D) [10,20,30]

Answer: B) [20,30]

12. Negative indexing starts from:

A) Beginning
B) End
C) Middle
D) Random position

Answer: B) End

13. Which method sorts a list?

A) order()
B) sort()
C) arrange()
D) sequence()

Answer: B) sort()

14. Which operation combines two lists?

A) +
B) -
C) *
D) /

Answer: A) +

15. What is the output of (1,2,3)[0]?

A) 0
B) 1
C) 2
D) 3

Answer: B) 1

16. Dictionary values are accessed using:

A) Index
B) Key
C) Slice
D) Loop

Answer: B) Key

17. Which method returns dictionary keys?

A) keys()
B) values()
C) items()
D) getkeys()

Answer: A) keys()

18. Which method returns dictionary values?

A) keys()
B) values()
C) items()
D) get()

Answer: B) values()

19. Which method removes all items from a dictionary?

A) remove()
B) delete()
C) clear()
D) popall()

Answer: C) clear()

20. Strings in Python are:

A) Mutable
B) Immutable
C) Numeric
D) None

Answer: B) Immutable

21. Which operator concatenates strings?

A) *
B) +
C) /
D) %

Answer: B) +

22. Output of "Python".upper()?

A) python
B) PYTHON
C) Python
D) Error

Answer: B) PYTHON

23. Output of "HELLO".lower()?

A) HELLO
B) hello
C) Hello
D) Error

Answer: B) hello

24. Which method replaces a substring?

A) replace()
B) change()
C) swap()
D) modify()

Answer: A) replace()

25. Which method removes extra spaces from both ends?

A) trim()
B) strip()
C) clean()
D) remove()

Answer: B) strip()

26. NumPy stands for:

A) Numerical Python
B) Number Python
C) Numeric Program
D) New Python

Answer: A) Numerical Python

27. Which library is imported as np?

A) Pandas
B) NumPy
C) Matplotlib
D) CSV

Answer: B) NumPy

28. Which function creates a NumPy array?

A) np.make()
B) np.create()
C) np.array()
D) np.list()

Answer: C) np.array()

29. What is the output?

```
import numpy as np
a=np.array([1,2,3])
print(a+1)
```

A) [1,2,3]
B) [2,3,4]
C) [3,4,5]
D) Error

Answer: B) [2,3,4]

30. Which operation multiplies each element by 2?
a=np.array([1,2,3])

A) a+2
B) a-2
C) a*2
D) a/2

Answer: C) a*2

31. What is the output of np.zeros(3)?

A) [0 0 0]
B) [1 1 1]
C) [3 3 3]
D) Error

Answer: A) [0 0 0]

32. CSV stands for:

A) Common Standard Values
B) Comma Separated Values
C) Character Separated Values
D) Column Standard Values

Answer: B) Comma Separated Values

33. Which module is used to work with CSV files?

A) math
B) os
C) csv
D) random

Answer: C) csv

34. Which mode opens a file for reading?

A) r
B) w
C) a
D) x

Answer: A) r

35. Which mode opens a file for writing?

A) r
B) w
C) a
D) rb

Answer: B) w

36. Which mode appends data to a file?

A) r
B) w
C) a
D) x

Answer: C) a

37. Which function reads CSV data in Pandas?

A) pd.load_csv()
B) pd.read_csv()
C) pd.import_csv()
D) pd.open_csv()

Answer: B) pd.read_csv()

38. Which function saves data to a CSV file in Pandas?

A) write_csv()
B) save_csv()
C) export_csv()
D) to_csv()

Answer: D) to_csv()

39. Structured healthcare datasets are organized into:

A) Rows and Columns
B) Pictures
C) Audio Files
D) Videos

Answer: A) Rows and Columns

40. Each row in a healthcare dataset generally represents:

A) Doctor
B) Hospital
C) Patient Record
D) Disease

Answer: C) Patient Record

41. Patient ID is an example of:

A) Record
B) Attribute/Column
C) File
D) Dataset

Answer: B) Attribute/Column

42. Which library is commonly used for healthcare data analysis?

A) Pandas
B) NumPy
C) Both A and B
D) None

Answer: C) Both A and B

43. Pharmaceutical datasets may contain:

A) Drug Name
B) Price
C) Stock Quantity
D) All of the Above

Answer: D) All of the Above

44. Which command displays the first five rows of a dataset?

A) info()
B) head()
C) show()
D) first()

Answer: B) head()

45. Which command gives information about dataset columns and data types?

A) details()
B) info()
C) show()
D) shape()

Answer: B) info()

46. Which command selects the "Price" column?

A) data(Price)
B) data["Price"]
C) data{Price}
D) data

Answer: B) data["Price"]

47. Which operation is used to filter records?

A) Conditional Selection
B) Looping
C) Printing
D) Indexing Only

Answer: A) Conditional Selection

48. Which function calculates the average of a numerical column in Pandas?

A) avg()
B) average()
C) mean()
D) median()

Answer: C) mean()

49. Which operation arranges dataset values in order?

A) Filtering
B) Sorting
C) Indexing
D) Grouping

Answer: B) Sorting

50. The main purpose of importing pharmaceutical datasets into Python is:

A) Gaming
B) Data Analysis and Managemen

C) Web Browsing
D) Graphic Design

Answer: B) Data Analysis and Management

Short Questions with Answers

1. What is a list in Python?

Answer: A list is an ordered, mutable collection of elements enclosed in square brackets [].

2. What is a tuple?

Answer: A tuple is an ordered, immutable collection of elements enclosed in parentheses ().

3. What is a dictionary?

Answer: A dictionary is a collection of key-value pairs enclosed in curly braces {}.

4. What is indexing?

Answer: Indexing is the process of accessing elements using their position number.

5. What is slicing?

Answer: Slicing is used to extract a portion of a sequence such as a list or string.

6. Which method is used to add an element to a list?

Answer: append()

7. Which method is used to remove an element from a list?

Answer: remove()

8. What is the index of the first element in a Python list?

Answer: 0

9. How are dictionary values accessed?

Answer: By using their keys.

10. What is string concatenation?

Answer: Joining two or more strings using the + operator.

11. Which method converts a string to uppercase?

Answer: upper()

12. Which method converts a string to lowercase?

Answer: lower()

13. What is NumPy?

Answer: NumPy is a Python library used for numerical and array operations.

14. Which function is used to create a NumPy array?

Answer: np.array()

15. What does CSV stand for?

Answer: Comma Separated Values.

16. Which mode is used to read a file?

Answer: r

17. Which mode is used to write a file?

Answer: w

18. What is a structured healthcare dataset?

Answer: A healthcare dataset organized into rows and columns.

19. Which Pandas function is used to read a CSV file?

Answer: pd.read_csv()

20. What is the purpose of importing pharmaceutical datasets into Python?

Answer: To analyze, manage, and manipulate pharmaceutical data efficiently.

Long Questions

1. Explain Lists, Tuples, and Dictionaries with examples.
2. Describe Indexing and Slicing in Python with examples.
3. Explain String Manipulation Techniques in Python.
4. What is NumPy? Explain array creation and arithmetic operations.
5. Explain Reading and Writing CSV Files in Python.
6. What are Structured Healthcare Datasets? Discuss their importance.
7. Explain importing pharmaceutical datasets and accessing data using Pandas.
8. Discuss data manipulation operations on pharmaceutical datasets.

9. Explain basic operations on Lists and Dictionaries with examples.
10. Write a detailed note on Data Structures and File Handling in Python and their applications in healthcare.

CASE-BASED LEARNING (CBL)

Case Study 1: Managing Medicine Inventory Using Lists

Scenario

A pharmacy stores medicine names in a list. The pharmacist wants to add a new medicine and remove an expired medicine.

Python Code

```
</>python
medicines = ["Paracetamol", "Amoxicillin", "Cetirizine"]
# Add a medicine
medicines.append("Metformin")
# Remove expired medicine
medicines.remove("Cetirizine")
print(medicines)
```

Q1. What is the purpose of this program?
Answer: To manage medicine inventory using a list.
Q2. Which data structure is used in the program?
Answer: List.
Q3. What is the name of the list variable?
Answer: medicines
Q4. How many medicines are initially stored in the list?
Answer: Three.
Q5. Which medicines are initially present in the list?
Answer: Paracetamol, Amoxicillin, and Cetirizine.
Q6. Which method is used to add a new medicine?
Answer: append()
Q7. Which medicine is added to the list?
Answer: Metformin.
Q8. Which method is used to remove a medicine?
Answer: remove()
Q9. Which medicine is removed from the list?
Answer: Cetirizine.
Q10. What is the final list after adding and removing medicines?
Answer: ['Paracetamol', 'Amoxicillin', 'Metformin']

Case Study 2: Patient Information Using Dictionary

Scenario

A hospital stores patient information in a dictionary.

Python Code

```
</>python
patient = {
   "ID": "P101",
   "Name": "Ravi",
   "Disease": "Diabetes"
}
print(patient["Name"])
```

Output

```
</>python
Ravi
```

Q1. What is the purpose of this program?
Answer: To store and access patient information using a dictionary.
Q2. Which data structure is used in the program?
Answer: Dictionary.
Q3. What is the name of the dictionary variable?
Answer: patient
Q4. What key stores the patient ID?
Answer: "ID"
Q5. What key stores the patient name?
Answer: "Name"
Q6. What key stores the disease information?
Answer: "Disease"
Q7. What is the value of the key "Name"?
Answer: Ravi
Q8. How is the patient's name accessed from the dictionary?
Answer: patient["Name"]
Q9. What is the output of the program?
Answer: Ravi
Q10. How many key-value pairs are stored in the dictionary?
Answer: Three.

PROBLEM-BASED LEARNING (PBL)

Problem 1: Find Medicines with Low Stock

Problem

A pharmacy wants to identify medicines with stock less than 50 units.

Dataset

Medicine	Stock
Paracetamol	100
Amoxicillin	40
Metformin	30

Solution

```
</>python
import pandas as pd

data = {
   "Medicine":["Paracetamol","Amoxicillin","Metformin"],
   "Stock":[100,40,30]
}
df = pd.DataFrame(data)
low_stock = df[df["Stock"] < 50]
print(low_stock)
```

Expected Result

```
  Medicine      Stock
1 Amoxicillin   40
2 Metformin     30
```

Q1. What is the problem in this case study?
Answer: To identify medicines with stock less than 50 units.
Q2. Which library is used in the program?
Answer: Pandas.
Q3. How is Pandas imported?
Answer: import pandas as pd
Q4. What is the name of the DataFrame variable?
Answer: df

Q5. Which column contains medicine names?
Answer: "Medicine"

Q6. Which column contains stock values?
Answer: "Stock"
Q7. What condition is used to find low-stock medicines?
Answer: df["Stock"] < 50

Q8. What is stored in the variable low_stock**?**
Answer: Medicines with stock less than 50 units

Problem 2: Calculate Average Drug Price

Problem

Find the average price of medicines.

Solution

```
</>python
import pandas as pd
data = {
   "Medicine":["A","B","C"],
   "Price":[50,80,70]
}
df = pd.DataFrame(data)
print(df["Price"].mean())
```

Output

```
</>python
66.67
```

Q1. What is the objective of this program?
Answer: To calculate the average price of medicines.
Q2. Which library is used in the program?
Answer: Pandas.
Q3. How is Pandas imported?
Answer: import pandas as pd
Q4. What is the name of the DataFrame variable?
Answer: df
Q5. Which column contains the medicine prices?
Answer: "Price"
Q6. What prices are given in the dataset?
Answer: 50, 80, and 70.
Q7. Which function is used to calculate the average?
Answer: mean()

Q8. What does df["Price"].mean() calculate?
Answer: The average price of all medicines.