Date / /20

List [ ]	Tuple ( )	Set { }	Dictionary { }
A list is a collection of ordered elements	A tuple is a sequence of elements separated by commas & enclosed in parentheses	A set is a built-in data structure in python that represents a collection of unique elements.	A dictionary is a collection of key-value pairs where each key is unique & associated with a value.
Lists maintain the order of the elements they contain	Tuples maintain the order of the elements they contain	Sets do not maintain	Dict do not maintain
List can be accessed by [ ] Index	Tuples can be accessed by Index	Sets cannot	Dict cannot
Lists can be modified by adding or removing elements	Tuples cannot be modified by adding or removing elements	Sets can	Dict cannot
Lists can contain duplicate elements	Tuples can contain duplicate	Sets cannot	Dict cannot

⟹ List :- A list is a collection of ~~Tuple :-~~ elements. Lists can be of different data types such as ~~Pigure~~, float, strings etc. various operations such as adding or removing elements or searching & sorting can be performed in a list.

Eg :- my_list = ~~[1, 2, 3, 4, 5]~~ [1, 2, 3, 4, 5, "hello"]
~~my_list = [1, 2, 3, 4, 5]~~

my_list[2] = 6 (changing item)

my_list.append(7) (adding item at end)

my_list.remove(4) (removing item)

print(my_list)

Output :- [1, 2, 6, 5, 7]

⟹ Tuple :- A tuple is a sequence of elements separated by commas & enclosed in parentheses. Tuples are similar to lists, but they cannot be modified once created. This means that you cannot add, remove or modify in tuple.

Eg :- my_tuple = (1, 2, 3, 4, 5)

~~print(Python~~ print(my_tuple[:3])

Output :- (1, 2, 3)

⟹ Dictionary :-

A dictionary in Python is a collection of key-value pairs, where each key is unique and associated with a value. We can change the values of a dictionary. They are useful for storing & accessing data.

~~Eg :- {"Fruits" : "Apple" "Cars" : "BMW"}~~

Eg :- countries = {"India" : "NewDelhi",
"Japan" : "Tokyo"}

print(countries['India'])

print(countries['Japan'])

Output :- New Delhi
Tokyo

Concatination means + adding!!

print ("Hello" + "World")

Output :- HelloWorld

Hi*3 ⇒ HiHiHi

(Duplication *)

format() → To insert values

name = Sun

Age = 25

print ("My name is {} & age is {}." format (name, age))

String Methods (all 6 imp)

① strip() → Remove spaces

"Hello ".strip() → "Hello"

② lower() → Converts to lowercase

"HELLO".lower() → "hello"

③ upper() → Converts to uppercase

"hello".upper() → "HELLO"

④ replace(a,b) → replaces a with b

"apple".replace("a","b") → "bpple"

Date / /20

⑤ split(",") → splits by , into a list
"a,b,c".split(",") → ['a','b','c']

⑥ startswith("Py") → checks if starts with Py
"Python".startswith("Py") → True

Lists []	Tuples ()	Sets {}	{} Dictionaries
→ Ordered	→ Ordered	→ Unordered	→ Unordered
→ mutable	→ Immutable	→ mutable	→ Indexed by keys
→ Allow duplicates	→ Allow duplicates	→ unique elements only	→ Key-value pairs

Slicing :- a tuple

```
my_tuple = (1, 2, 3, 4, 5)
print(my_tuple[1:3])
```

→ Output :- (2, 3)

File → File handling allows reading & writing data to disk files. It provides a way to store data permanently, unlike standard input/output which is temporary.

Types of Files → Common types include text, image, audio, video, binary & CSV files.

Operations → You can open, read, write & close files in Python.
Properly closing files frees resources.

To open file → open()

f = open ("my.txt", "r")

'r' → Read mode (default)

'w' → write mode (overwrites existing content)

'a' → append mode (adds to existing content)

'rb' / 'wb' → read & write binary mode

Eg:- f = open ("my.txt", "w")

f.write("Hello, world,")

f.close()

CSV files :

A CSV file stores ~~short~~ structured data where values are separated by commas.

⇒ To read a CSV file

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

⇒ To write to a CSV file

```
import csv
with open ('data.csv', 'w') as file
    writer = csv.writer (file)
    writer.writerow (["Name","Age"])
    writer.writerow (["Sun", 22])
```

→ Python is an object oriented programming language where almost everything is an object.

Date / /20

Class 1.1

A class in python is a blueprint for creating objects. It defines properties and behaviours that objects instantiated from the class will have.

Eg :-
```
class Person :
    def __init__(self, name, age):
        self.name = name
        self.age = age
```
→ used for initializing object properties when object is created

Inheritance allows one class (child class) to inherit attributes & methods from another (Parent class)

```
class Parent :
    def greet(self):
        print("Hello")
class child(Parent):
    pass
obj = child()
obj.greet()
```
output : Hello.

Operator overloading

Overloading operators like +, *, etc for custom object behavior.

Eg :- + operator for objects

P.T.O

```
class A:
    def __init__(self, value):
        self.value = value

    def __add__(self, other):
        return self.value + other.value
ob1 = A(10)
ob2 = A(20)
print(ob1 + ob2)
```

Output : 30