Sequence (String, List, tuple)

"A string is a collection of one or more character put in a single quote, double-quotes or triple quote."
It is represented by str class.

(2) List :>

Lists are like the arrays, declared in other languages which is a ordered collection of data. List is very flexible as the items in a list do not need to be of the same type.

* "List is a collection of things, enclosed in ([ ]) Square brackets and separated by commas." [ ]

(3) Tuple :>

Tuples is also an ordered collection of python objects. The only differences between tuple and list is that tuple are immutable.
tuple cannot be modified after it is created. It is represented by 'tuple' class.

⇒ Note :> Tuple can also be created with a single element, but it is a bit tricky. Having one element in the parentheses is not sufficient, there must be a trailing 'comma' to make it a tuple. ( )

Boolean, Sets.

M.Imp

* (iii) **Boolean** :-> Boolean is one of the built-in data types provided by python, which represents one of the two values. i.e. 'True' or 'false'.
"It is used to represent the truth values of expression.".
It is denoted by the class bool.

-> **Note** :-> True and false with Capital 'T' and 'f' are valied booleans otherwise python will throws an error.

M.Imp

* (iv) **Set** :-> Set is an unordered collection of data type ~~is~~ that is iterable, mutable and has no duplicate element.
The order of elements in a set is undefined through it may consist of various element.

Creating a Set :-
Sets can be created by using the built-in Set( ) function with an itrable object or a sequance by placing the sequance inside ({}) curly braces, separated by 'comma'.

Dictionary

m. Imp

x(v) **Dictionary** :-> Dictionary in python is an unordered collection of data values, used to store data values like a map, which unliked other Data Types that hold only single values as an elemend, Dictionary hold 'key : value' pair.
Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a Colon : , whereas each key is separated by a 'comma'.
Dictionary can be created by placing a sequance of element within curly {} braces, seprated by 'comma'.

Note :-> Dictionary keys are case sensetive, same name but different case of keys will be treated distinctly.

* **Indentation** :-> Indentation is a very important concept of python because without properly indenting the python code, you will ~~see~~ end up seeing IndentationError and the code will not get compiled.
"Python indentation refers to adding white space before a statement to a particular block of code.".

Imp * **Comments** :-> Comments are the lines in the code that are ignored by interpreter during the execution of the program.
Comment enhance the readablity of the code and help the programmers to understand the code very carefully.

Types of comments in python :->
There are three types of comments in python programming language :-

- Single line comments.
- Multiline comments.
- Docstring comments.

* (i) Single line comment :-> A single line comment starts and ends in the same line. We use the # symbol to write a single-line comment.

For Example :->

```
# Creat a variable
name = 'Shivam maurye'
# Print the value
Print (name).
```

Output :->

Shivam maurya.

* (ii) Multiline Comments :->

Python does not provide the option for multiline comments. How-ever, there are ~~can~~ different ways through which we can write multiline comments.

(i) Using multiple Hashtags (#).
(ii) Using String Literals.

(i) Using multiple Hashtags (#) :->

We can use multiple hashtags (#) to write multiline comments in python. Each line consider as single line comment.

Ex:⇒

```
# Python program to demonstrate
# multiline comments.
Print("multiline comments")
```

Output:⇒

multiline comments.

(ii) Using string literals:⇒

Python ignores the string literals that are not assigned to a variable so we can use these string literals as a comments.

Example:⇒

```
""" Python program to demonstrate
multiline comments """.
Print ("multiline comments.")
```

Output:⇒ multiline comments.

(3)* Docstring:⇒ Docstring is the string literals with triple quotes that are appeared right after the function. It is used to associate documentation that has been written with modules, functions, classes, and methods.

It is added right below the functions, modules, or classes to describe what they do. In python, the docstring is then made available via the __doc__ attribute.

Example :=>

```
def multiply (a, b):
    """ multiplies the value of a & b"""
    Return a * b
# Print the docstring of multiply
    function.
Print(multiply.__doc__)
```

Output :=>

multiplies the values of a & b.

m. imp

* Type Conversion :->

Type conversion is the conversion of an object from one data type to another data type.

Types of Type Conversion :->

There are two types of Type Conversion in python programming :—

(i) Implicit Type conversion.
(ii) Explicit Type conversion.

(i) Implicit Type Conversion :->

It is done automatically by the python interpreter.

(ii) Explicit Type conversion :->

Needs to be done manually by the programmer.

Ex :->

```
x = 1
Y = 2.8
z = 1J
a = float(x)
b = int(Y)
c = complex(x)
Print(a)
Print(b)
Print(c)
```

Output :->

```
a = 1.0
b = 2
c = 1+0j
```

m.Imp

⇒ Control flow statement in python :-

A control flow is the order in which the program's code executes.

The control flow of a program is regulated by conditional statement, loops, and function calls.

There are three types of control structures :-

→ (i) Sequential :- Default mode.
→ (ii) Selection :- Used for decision and branching.
→ (iii) Repetition :- used for looping.

* (i) Sequential statement :-

Sequential statement are a set of statements whose execution process happens in a sequance. The problem with sequential statements is that if the logic has broken in any one of the lines, them the complet source code execution will break.

Example :-

```
a = 20
b = 10
c = a - b
Print("Subtraction is :", c)
```

Output :-

```
Subtraction is : 10
```

Conditional Statements

* (ii) Selection / Decision control statements :→

The Selection statement allows a program to test several conditions and execute instructions based on which condition is true.

Some Decision control stedements are :→

- Simple if.
- if-else.
- nested if.
- if-elif-else.

(i) Simple if :→ A simple if only has one condition to check.

Example :→

```
n = 10
if n % 2 == 0:
    Print(" n, " is an even number")
```

Output :→

10 is an even number.

(ii) if-else statement :→

If-else statement is used to execute both the true part and the false part of a given condition. if the condition is true, if block code is executed and if the condition is false,

the else block of code is executed.

Syntax :-

```
if (condition) :
    Statement
else :
    Statement.
```

Program/Example :-

```
n = 10
if n % 2 == 0 :
    Print(n," is an even number.")
else :
    Print(n," is odd number.")
```

Output :-

10 is an even number.

(iii) Nested if statement :-

Nested if statements are an if statement inside another if statement.

Syntax :-

```
→ if test condition :
→     if test condition :
          Statement
      else :
          Statement
      elif condition :
          Statement
      else :
          statement
```

Example :->

```
a = 5
b = 10
c = 15
if a > b :
    if a > c :
        Print(a," is greater.")
    ~~elif b > c~~ :
    else :
        Print(c," is greater.")
    elif b > c
        Print(b," is greater.")
    else :
        Print(c," is greater.")
```

(iv) if-elif-else statement :->

The if-elif-else statement is used to conditionally execute a statement or a block of statement.

Syntax :->

```
if condition :
    Statement
elif condition
    Statement
else :
    Statement
```

Example :-

```
x = 15
if x > 0:
    Print(x," is positive number.")
elif x == 0:
    Print(x," is neutral number.")
else:
    Print(x," is negative number")
```

Output :-

15 is positive number.

*(iii) Repeatition :-

A repetition statement is used to repeat a group (block) of programming instructions

There are two loops / repetitive statement in python :-

- for loop.
- While loop.

Imp (i) for loop :- A for loop is used to iterate over a sequence that is eighter a list, tuple, dictionary, or a set.

Syndax, Example is End of the copy.

(ii) while loop :- " " " "

(iii) Nested loop :- " " " "

Unit = III

Python :->

(i) **file** :-> A file is some information or data which stays in the computer storage devices.
"A file is a container in computer storage devices used for storing data."

* **Types of file in python** :->
There are two types of file in python :-
(i) Text file.
(ii) Binary file.

(i) **Text file** :-> A Text file is a type of digital, non executable file that contains latters, numbers, symbols or a combinations.

(ii) **Binary file** :-> A Binary file is a file whose content is in a binary format consisting of a series of sequential bytes, each of which is eight bits in length.

* **Operations of file** :->
There are mainly 3 operations of file :-

(i) Open file.
(ii) Read or write.
(iii) Close the file.

(i) Opening files in python :-
In python we can open file using open() method. Suppose we have a file named test.txt. with some content.

Syntax :-

```
file_object = open("filename")
```

Example :-

```
file1 = open("test.txt")
```

⇒ To read the file :-

Syntax :-

```
file_object = Open("file_name", "Access_mode")
```

Example :-

```
file1 = open("test.txt", "r")
```

⇒ To write in the file :-

Syntax :-

```
file_object = Open("filename", "Access_mode")
```

Ex :-

```
file1 = open("test.txt", "w").
```

(ii) Reading files in python :->

After open a file, we can read the content using read() method.

for example :->

```
# open a file
file1 = open("test.txt", "r")
# read the file.
read_content = file1.read()
Print(read_content).
```

Output :->

```
This is a test file
Hello from the test file.
```

(iii) writing to files in python :->

To write into a file, we need to open it in write mode by passing "w" inside open() as a second argument.

Suppose we don't have a file named test1.txt. Let's see what happend if we write content to the test1.txt file

Example :->

```
with open('test1.txt', 'w') as file2:
# write contents to the test1.
txt file.
```

file2.write ('Programming is fun.')
file2.write ('program for beginners.')

Output :->

Programing is fun.
Program for beginners.

⇒ Here, new test1.txt file created with the contents specified inside the write() method.

(IV) Closing files in python :-

we can close the file by using close() method.

Ex :->

```
file1 = open("test.txt", "r")
read_content = file1.read()
Print(read_content)
#close the file.
file1.close()
```

Output :->

This is a test file.
Hello from the test file.

⇒ method of file :->

(i) close() :- to close an opened file.
(ii) Read(n) :- Read at most n char from the file.
(iii) Readlines(n = -1) :- Read and returns a list of lines from the file.

Object and class

⇒ Object oriented programming (Python):

⇒ **Class** :→ A class is considered as a blue print of objects. which contain all the information of a object.

⇒ **Define Python class** :→
We can use the 'class' keyword to creat a class in python.

Syntax :→

```
class ClassName:
```

Example :→

```
Class ClassName:
```

Here, we have created a class named 'ClassName'.
Let'sgee an Example :—

```
Class Bike:
    Name = " "
    gear = 0
```

Here :—

- Bike :- the name of the class.
- Name / gear - variablers inside the class with default value (" ") and (0).

Note :→ The variable inside a class called attribute

⇒ **Object :→** An object is called an instance of a class.

Suppose Bike is a class then we can creat object like bike1, bike2, etc. from the class.

Syntax :→

```
object_Name = class_Name().
```

Example :→

```
# creat a class
Class Bike :
    Name = " "
    gear = 0
# creat object of class.
bike1 = Bike().
```

Here, bike1 is the object of the class. Now, we can use this object to access the Class Attribute.

Access class Attributes using Object :→

we can use the (.) notation to access the attributes of a class.

(P.T.O)

for example :->

```
# modify the Name attribute
bike1.name = "Hero Bike".

# access the gear attribute.
bike1.gear = 11
```

Here, we have used 'bike1.name' and 'bike1.gear' to change and access the value of 'Name' ang 'gear' attribute.

Example of class and object :-

```
Class Bike:
    Name = "  "
    gear = 0
Bike1 = Bike()
Bike1.gear = 11
Bike1.Name = "Hero Bike"
Print(f"Name: {Bike1.Name},
        Gears: {Bike1.gear}")
```

Output :->

Name: HeroBike, Gears: 11

(i) Constructor :-

Constructor in python is a special method that is called when an object is created.

→ The purpose of a python construc-tor is to assign values to the data members within the class when an object is initialized.

→ The Name of the constructor is always __init__ .

Example :-

```
Class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age.
Person = Person("John", 30)
Print(Person.name)
Print(Person.age)
```

Output :-

John
30

Types of Constructor :-

There are 3 types of Construction in python.