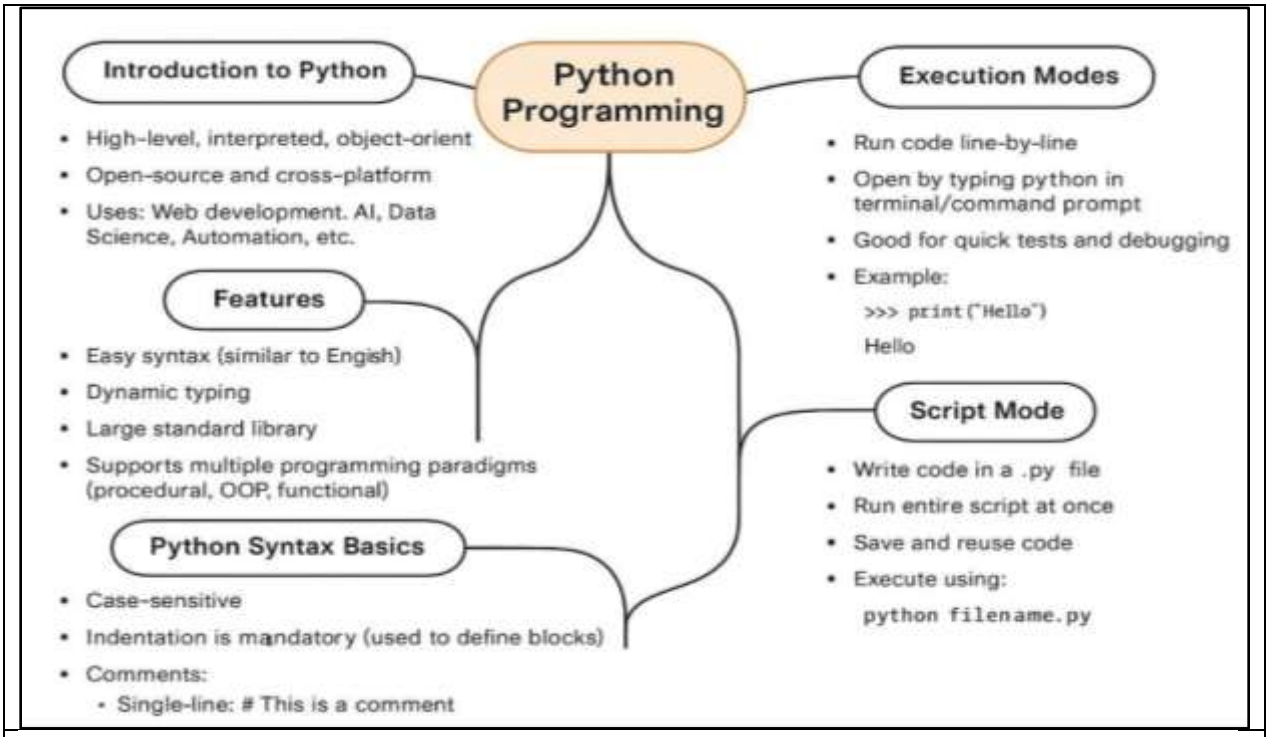
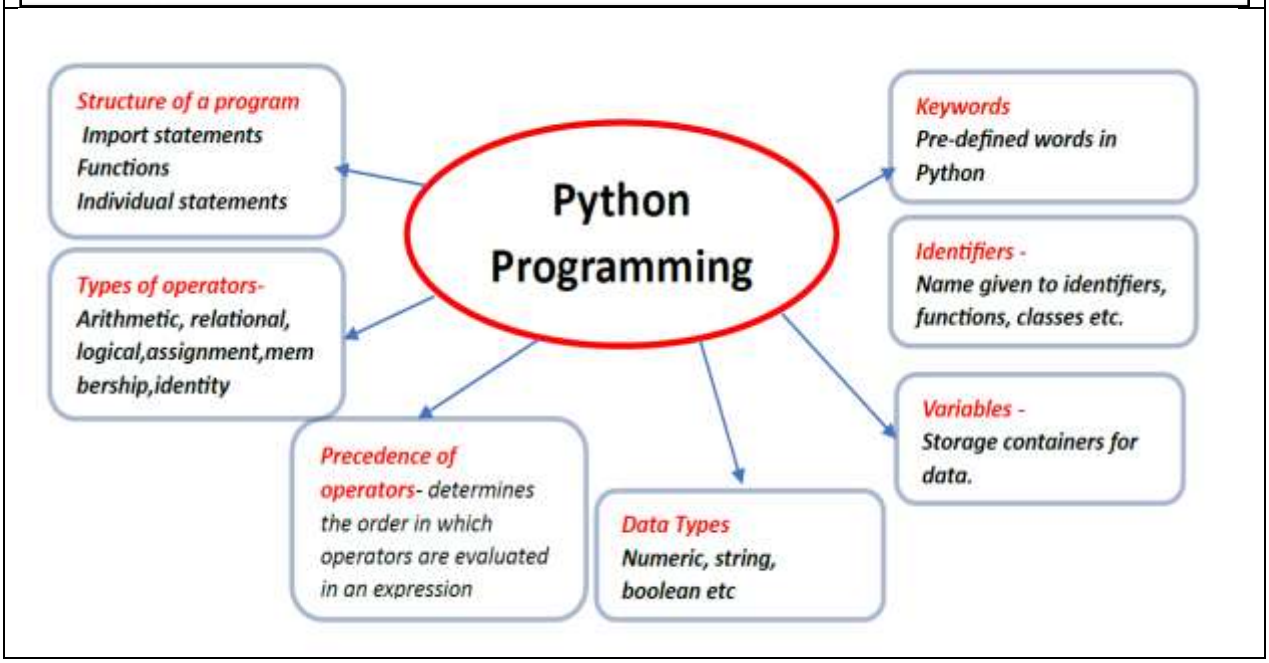
Unit 2: Introduction to Python

BASICS OF PYTHON

Mind Map





1. Basics of Python Programming & Execution Modes (Interactive and Script Mode)

Python Basics:

Python is a high-level, interpreted, and general-purpose programming language. It emphasizes readability and ease of use, making it a great choice for beginners. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

2. Execution Modes:

Python can be run in two primary execution modes:

- **Interactive Mode:**
  - You enter individual Python commands or expressions, and the interpreter immediately executes and displays the result.
  - This mode is great for quick testing and exploring code snippets.
  - **Example:** Open the Python shell and type print("Hello, World!") to immediately get the result.

    In Interactive mode, commands are given in front of Python command prompt >>>

    Example

    >>> 10+ 15
    25
    >>> print ('Hello, World')
    Hello, World

    Interactive Mode is **helpful for testing statement/code,** you type the commands, commands are executed one by one and get the result or error one by one.

- **Script Mode:**
  - Python program is written in a file(.py), which are then executed as a whole by the Python interpreter.
  - This mode is suited for developing larger programs or when you want to run a sequence of Python commands that are stored in a file.

    Steps for writing in Script mode –

    1. Open Python IDLE
    2. From File Menu Choose – New File
    3. Type the codes on File named Untitled
    4. Click on Save from file menu

Interactive Mode Vs Script Mode

In Interactive Mode, the commands or statements entered by the user are not saved. However, in Script Mode, the program code or statements are first saved in a .py file, and then the code is executed.

3. The Structure of a Program

The structure of a Python program typically consists of the following elements:

- **Modules**: Python code is often organized into modules, which are files containing Python definitions and statements.
- **Functions**: Functions are defined using the def keyword and are used to encapsulate code that performs specific tasks.
- **Statements and Expressions**: A statement in Python represents an action (e.g., print ()), while an expression computes a value (e.g., x + y).
- **Control Flow**: Conditional statements (if, else, elif) and loops (for, while) direct the flow of execution.

A basic simple program structure might be look like this:

```
print (“hello”)
```

A basic program structure using function might look like this:

```
def my_function():                    #defining of function
   # Function code here
   print(“hello”)

 # Main program execution
   my_function()                      #calling of function
```

4. Indentation

In Python, indentation is critical and is used to define the structure of code blocks. Unlike many other programming languages, Python does not use braces {} to define blocks of code (like loops or functions). Instead, it uses whitespace indentation.

- **Consistent Indentation**: Python requires consistent indentation throughout a block of code. A common convention is using 4 spaces for each level of indentation.
- **Error if Inconsistent**: If you mix spaces and tabs, or if the indentation is inconsistent, Python will throw an IndentationError.

Example:

```
if  a==0:
     print("Hello, World!")  # print statement part of if condition
print(“Hello, World!”) # print statement not a part of if condition
```

5. Identifiers

Identifiers in Python are names used to identify variables, functions, classes, and other objects.

- **Rules for Identifiers:**
  - Must begin with a letter (A-Z or a-z) or an underscore (_).
  - Can be followed by letters, digits (0-9), or underscores.
  - Case-sensitive: Schoolname, SchoolName, and SCHOOLNAME all three are different identifiers in python.

- Cannot be a keyword (e.g., if, else, def, etc.).

Example: my_variable = 10

Valid Identifiers: Rollno, ROLLNO, ROLL_NO, RNO92A, _RNO, RFILE
Invalid Identifiers: DATA_REC, break(keyword), Roll.No, 19Rno, Roll No

6. Keywords

Keywords are reserved words in Python that have special meaning and cannot be used as identifiers.

Example: if, else, elif, for, while, def, class, import

7. Constants

Values that do not change during execution of program are called constants. In Python, there are no built-in constants (values that cannot be changed once set). However, it's a common convention to use **uppercase letters** for constant-like values, signaling that they should not be changed.

- **Example:** PI = 3.14159 MAX_USERS = 1000

8. Variables

A variable in Python is an identifier whose value can be changed. Python is a **dynamically typed** language, meaning the type of a variable is determined at runtime, based on the value it holds.

- **Declaring Variables**: You don't need to specify the type of the variable; Python automatically assigns the appropriate type based on the value.
  - **Example:**
    - age = 25 # This assigns an integer value to 'age'
    - name = "John" # This assigns a string value to 'name'
- **Reassigning Variables**: Python allows the reassignment of variables to different data types.

```
age = 25                # 'age' contains value of integer type
age = "twenty-five"     # now 'age' contains value of string type
```

9. Types of Operators

Python supports various types of operators to perform operations on variables and values:

- **Arithmetic Operators (**+, -, *, /, //, %, **): Used for mathematical calculations.

Operator	**Description**	**Example** **X=10, y=20**
+ Addition	To adds values on either side of the operator.	>>>X + y >>>30
- Subtraction	Subtracts right hand operand from left hand operand.	>>>X – y >>>-10
* Multiplication	Multiplies values on either side of the operator	>>>X * y >>>200
/ Division	Divides left hand operand by right hand operand	>>>X / y

		>>>2
// Floor Division	The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) −	>>>9//2 >>>4 ---------- >>>9.0//2 >>>4.0
% Modulus	Divides left hand operand by right hand operand and returns remainder	>>>X % y >>>0
** Exponent	Performs exponential (power) calculation on operators	>>>X**2 >>>100

X variable in Capital letter and y variable in small letter

Note: Python is case sensitive This means that uppercase and lowercase letters are treated differently

Note: + and * operators are also used in String Manipulation

for e.g. X= 'Welcome' Y=2025

>>>X+Y

>>>Welcome2025

X='Welcome' Y=2

>>>X*Y

>>>WelcomeWelcome

- **Comparison Operators (**==, !=, >, <, >=, <=**)**: Used to compare two values. It either returns True or False according to the given condition.

Operator	**Description**	**Example X=20, y=10**
==	If values of two operands are equal, then the condition becomes true.	>>>(X == y) >>>**False**
!=	If values of two operands are not equal, then condition becomes true.	>>>(X != y) >>>> **True**
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	>>>(X> y) >>>**True**
<	If the value of left operand is less than the value of right operand, then condition becomes true.	>>>(X < y) >>>**False**
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	>>>(X >= y) >>>**True**

<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	>>>(X <= y) >>>False

Note: These operators can also be used for comparing string values, through the use of ASCII value of a String. ASCII value of A=65, Z=90, a=97, z=122

e.g. X= 'ABc' Y= 'Abc'

>>>X>Y # output will be False

- **Assignment Operators (**=, +=, -=, *=, /=,%=,**=,//=): Used to assign values to variables. It assigns values from right side operands to left side operand

Operator	Description	Example
=	Assigns values from right side operands to left side operand	Z = X + Y assigns value of X + Y into Z
+=	It adds right operand to the left operand and assign the result to left operand	X += Y is equivalent to X = X + Y
-=	It subtracts right operand from the left operand and assign the result to left operand	X -= Y is equivalent to X = X – Y
*=	It multiplies right operand with the left operand and assign the result to left operand	X *= Y is equivalent to X = X * Y
/=	It divides left operand with the right operand and assign the result to left operand	Y /= Z is equivalent to Y = Y / Z
%=	It takes modulus using two operands and assign the result to left operand	m %=n is equivalent to m = m % n
**=	Performs exponential (power) calculation on operators and assign value to the left operand	C **= a is equivalent to C = C ** a
//=	It performs floor division on operators and assign value to the left operand	c //= A is equivalent to c = c // A

- **Logical Operators (and, or, not)**: Used to combine conditional statements.

Operator	Description	Example **X=20, Y=10 Z=5**
and	Logical AND: True if both the operands are true otherwise False	X>Y and Y>Z **True**

or	Logical OR: True if either of the operands is true otherwise False	X>Y or Y<Z **True**	
not	Logical NOT: True if operand is False and vice versa	not (X>Y) **False** not(X>Y and Y>Z) **False**	

-

- **Membership Operators (in, not in):** Used to test membership in a sequence such as strings, lists, or tuples. There are two membership operators as explained below –

```
str= 'KVS 2025 Class IP'
Val='IP' in str
print(Val)
#output : True
```

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x **in** y, here in results, statement will be **True** only if x is a member of sequence y.
not in	Evaluates to true if it does not find a variable in the specified sequence and false otherwise.	x **not in** y, here in results, statement will be **True** if x is not a member of sequence y.

- **Identity Operators (is, is not):** Used to compare the memory location of two objects.
  is, is not are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal does not imply that they are identical.

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x **is** y, here it checks, if id(x) equals to id(y) then result is **True**
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x **is not** y, here it checks if id(x) is not equal to id(y) then result is **True**

- **Bitwise Operators(**&, |, ^, ~, <<, >>**)**: Used for bit-level operations.

10. Precedence of Operators

The precedence of operators determines the order in which operations are performed in an expression. Operators with higher precedence are evaluated before operators with lower precedence.

It is in descending order (upper group has higher precedence than the lower ones.)

Sr.No.	Operator	Description	Associativity
1	() **Highest Precedence**	Parentheses	Left to Right
2	**	Exponentiation (raise to the power)	Right to left
3	~ , + , -	Complement, unary plus and minus (method names for the last two are +@ and -@)	Left to Right
4	* , /, % , //	Multiply, divide, modulo and floor division	Left to Right
5	+ ,-	Addition and subtraction	Left to Right
6	>>, <<	Right and left bitwise shift	Left to Right
7	&	Bitwise 'AND'	Left to Right
8	^ , |	Bitwise exclusive `OR' and regular `OR'	Left to Right
9	<= < > >=	Comparison operators	Left to Right
10	<> == !=	Equality operators	Left to Right
11	= %= /= //= -= += *= **=	Assignment operators	Right to Left
12	not , or , and	Logical operators	Left to Right

Example:

```
result = 3 + 5 * 2 # Multiplication happens first
print(result)  # Output: 13
```

To change the order of evaluation, use parentheses.

Operator precedence and associativity of operators that decide the order in which parts of an expression are calculated. Precedence tells us which operators should be evaluated first, while associativity determines the direction (left to right or right to left) in which operators with the same precedence are evaluated.

11. Data Types

Data type identifies the type of data which a variable can hold and the operations that can be performed on those data.

Python supports a variety of built-in data types that can store values:

- **Numeric Types**:
  - int: Integer numbers (e.g., 5, -3).
  - float: Floating-point numbers (e.g., 3.14, -0.001).
- **Sequence Types**:
  - str: Strings (e.g., "Hello", 'Python').
  - list: Ordered, mutable collections (e.g., [1, 2, 3]).
  - tuple: Ordered, immutable collections (e.g., (1, 2, 3)).
- **Mapping Type**:
  - dict: Key-value pairs (e.g., {"name": "John", "age": 25}).
- **Boolean Type**:
  - bool: True or False.

12. Mutable and Immutable Data Types

- **Mutable Types**: These are data types/objects whose content or values can be changed after they are created.
  - Examples: list, dict.
  - These objects can be modified in place, so changes to a mutable object will affect all references to that object.
- **Immutable Types**: These are data types whose content or values cannot be changed after they are created.
  - Examples: int, str, tuple.
  - When an immutable object is modified, a new object is created, leaving the original object unchanged.

Example of Mutability:

```
a = [1, 2, 3]
b = a
a.append(4)   # Modifies the original list
print(b)      # Output: [1, 2, 3, 4]
print(a)      #Output:  [1, 2, 3, 4]
print(id(a))  # gives memory location of object
print(id(a))  # gives memory location of object
```

both a and b variable having same location in memory

Immutable object (string)

```
x = "hello"
y = x
x = "world"   # Creates a new string object
print(y)      # Output: hello (y is unaffected)
print(x)      # Output: world(x value changed)
print(id(x))  # gives memory location of object
print(id(y))  # gives memory location of object
```

both x and y variable having different location in memory

Multiple Choice Questions

1	Which function is used to display output in Python? A) show() B) print() C) display() D) output()
2	Identify the valid relational operator among the following. A) = B) += C) == D) and
3	Which of these is a valid Python identifier? A) 1var B) _var C) var-name D) var name

4	What is the correct way to assign a value to a variable in Python? A) x == 5 B) 5 = x C) x = 5 D) x := 5
5	Which operator is used for exponentiation in Python? A) ^ B) ** C) exp() D) %
6	Predict the output of the following code: print(3**2+55/11*-(-3+3)) A) -39.0 B) 27.0 C) -14.0 D) 9.0

Answer (Multiple Choice Questions)

1	B) print()
2	C) ==
3	B) _var
4	C) x = 5
5	B) **
6	D) 9.0

Assertion and Reasoning Questions

Choose correct option for given Assertion (A) and Reasoning (R)

A. Both A and R are true and R is the correct explanation of A.
B. Both A and R are true but R is not the correct explanation of A.
C. A is true but R is false.
D. A is false but R is true.

1	**Assertion (A) :**	List is mutable data type.
	Reasoning (R):	In-place change is not possible in list elements.
2	**Assertion (A) :**	1ABC is correct Identifier.
	Reasoning (R):	Identifier can't start with digits.
3	**Assertion(A):**	Variable names whether in capital or small letters are treated as different python.
	Reasoning(R):	Python is a case sensitive language.

Answer (Assertion and Reasoning Questions)

1	C) A is true but R is false
2	D) A is false but R is true
3	A) Both A and R are true and R is the correct explanation of A.

Very Short Questions with Answer

1	What is the None keyword in Python, and what does it represent?
	Ans. None is a special keyword in Python representing the absence of a value

	or a null value. It is often used to signify that a variable does not have any value assigned to it yet.
2	What is a variable in Python, and how is it different from a constant?
	Ans. A variable is a name that refers to a value that can be changed during the execution of the program, while a constant represents a value that remains unchanged throughout the program.
3	Differentiate between Keywords and Literals (Constants).
	Ans. Python **Keywords** are some predefined reserved words in Python that have special meaning. **Identifier** is a user-defined name given to a variable, function, class, module, etc. The identifier is a combination of character digits and an underscore. They are case-sensitive i.e., 'num' and 'Num' and 'NUM' are three different identifiers in python. It is a good programming practice to give meaningful names to identifiers to make the code understandable.
4	Arrange the precedence of operators from highest to lowest + , ** , // , ==
	Ans. precedence of operators from highest to lowest **, //, +, ==

Short Questions with Answer

1	**Nitin is trying to find out answers of these questions while finding output of a program. Help him to find out correct answers with reason.** **(i)** What will be output of **print(14%4)**? **(ii)** Identify the data type of **[3, 10]** **(iii)** What will be output of **print('5' + '5') ?** **(iv)** What will be output of **print(2**3**2) ?**
	Ans. **(i)** Output will be 2 Because % operator performs floor division and provides remainder after division. **(ii)** List as it is enclosed with [ ]. **(iii)** Output will be '55' Because + operator concatenates both the strings. **(iv)** Output will be 512, because multiple ** follows right associativity
2	Write a program to calculate simple interest.

	Ans:
	`P = int(input("Enter Principal : "))`
	`R = int(input("Enter Rate of Interest : "))`
	`T = int(input("Enter Time in Years : "))`
	`SI = (P*R*T)/100`
	`print("Simple Interest = ",SI)`

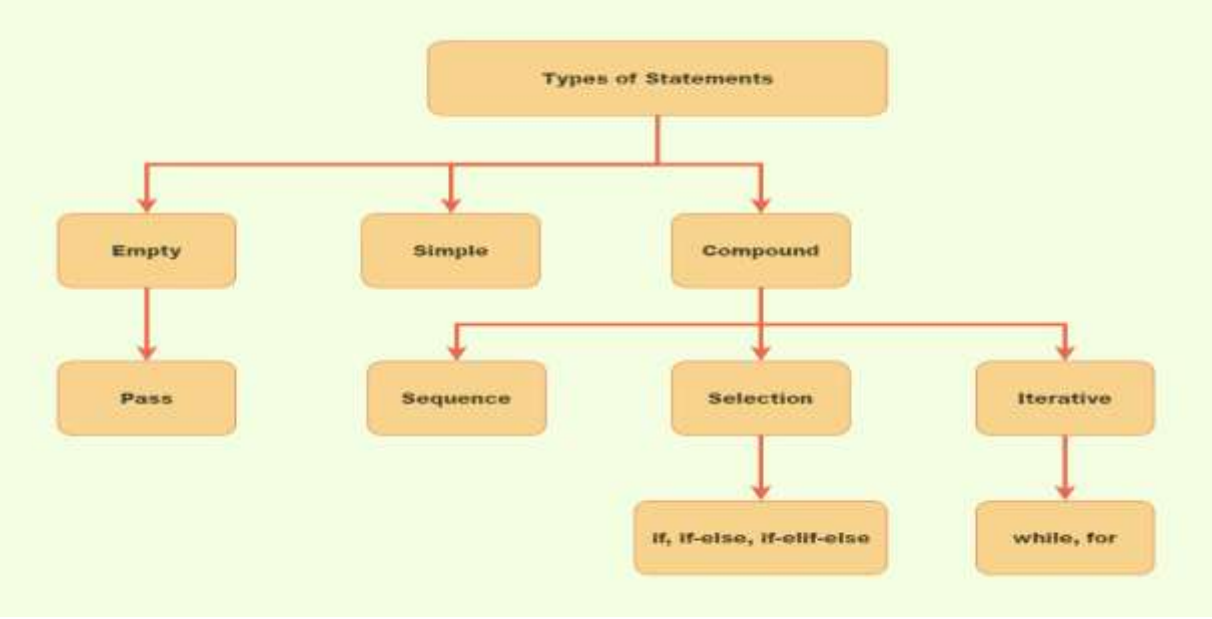
3	Write a Python program to find the average of three numbers where value of a is 10, b is 20 and c is 30.

	Ans:
	#To find the average of three numbers
	a = 10
	b = 20
	c=30
	avg = (a+b+c)/3
	print("The average is ",avg)

4	Predict Output – a, b, c = 2, 3, 4 a, b, c = a*a, a*b, a*c print(a, b, c)
	Ans. 4 6 8
5	Predict the Output – a, b, c = 10, 20, 30 p, q, r = c-5, a+3, b-4 print('a, b, c : ', a, b, c, end = '### ') print('p, q, r : ', p, q, r, sep = '$$ ')
	Ans: a, b, c : 10 20 30###p, q, r : $$25$$13$$16
6	Payal is working in a firm and given a task to find the average of two numbers x=7 y=5 print(x+y/2) she is not getting the correct average, give your suitable answer and rectify the code
	Ans: x=7 y=5 print**((x+y)/2)** **both x and y should be in parenthesis and both divided by 2 then she will be able to find correct average**

13. Statements

• Instructions executed by the Python interpreter.



- **Types of statements:**
  - **Empty statement: –** Statement which does nothing. It is as follows:

```
>>> pass
>>>
```

  - **Simple statement:** Any single executable statement is a simple statement.

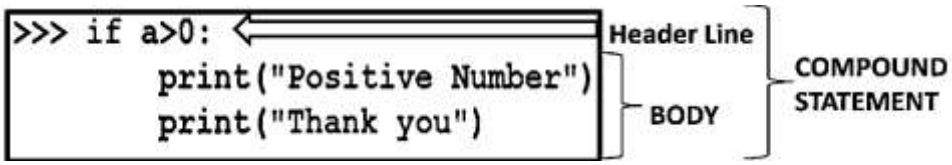
```
>>> a=10+20
>>> print(a)
```

  - **Compound statement:** A group of statements executed as a unit is a compound statement.

A compound statement has:

(i) a header line which begins with a keyword and ends with a colon.

(ii) a body consisting of one or more Python statements, each indented inside the header line. All the statements are at the same level of indentation.



14.Expressions and Evaluation

- Combinations of variables, operators, and values that yield a result.

- Examples:
  - o Arithmetic expressions: e.g. a=2 + 3
  - o Logical expressions: e.g. a **and** b , a **or** b
  - o String expressions: e.g. "Computer" **+** "Science"

- Can be part of a larger statement.

Evaluation

- The process of computing the result of an expression.
- Python evaluates expressions using the rules of precedence and associativity.

15. Comments

- Used to annotate code and make it more understandable.
  - **Single-line comments**: Begin with # symbol
    e.g., **#This is a comment.**
  - **Multi-line comments:** Enclosed in triple quotes
    e.g**. """ Multi-line comment**
    To give more description of code """

If the comment is more than one line then multi-line comment will be used

16. Input and Output Statements

- **Input:**

o input (): Reads a line of text input from the user (e.g., name = input ("Enter your name: ")).
o Always returns a string.

- **Output:**
  o print (): Outputs text or variables to the console (e.g., print ("Hello, World!")).
  o Can accept multiple arguments separated by commas (e.g., print ("Name:", name)).
  o Optional arguments like **sep** and **end** to customize the output
  e.g., print ("Hello", "World", **sep**="-")

17. Data Type Conversion

- Converting one data type to another.

- **Common functions:**
  o int(): Converts a value to an integer (e.g., int("42") result is 42)
  o float(): Converts a value to a float (e.g., float("3") result is 3.0)
  o str(): Converts a value to a string (e.g., str(42) result is "42")
  o list(): Converts a value to a list (e.g., list("abc") result is ['a', 'b', 'c'])
  o tuple(): Converts a value to a tuple (e.g., tuple([1, 2, 3]) result is (1,2,3))
  o dict(): Converts a value to a dictionary (when applicable, e.g., dict([('a', 1), ('b', 2)]) result is {'a':1 , 'b':2})

18. Debugging in Python

- **Definition:**
  o The process of finding and fixing errors or bugs in program code.

- **<u>Common Debugging Techniques:</u>**
  o print Statements: Insert **print ()** statements in your code to check values of variables and program flow

```
e.g.    print ("Checkpoint reached")   #output    Checkpoint reached

        x=15
        print ("Value of x:", x)        #output    Value of x: 15
```

19. Control Statements in Python

- **Sequence**: Sequence construct means statement are executed Sequentially:
- **Selection:** The Selection construct means the execution of statement(s) depending upon a condition-test.
- **Iterative:** Iteration construct means repetition of set of statements depending upon a condition test till the time of condition is true.

Understand concept of sequence, selection, iterative statement using real life example:
A child wants to celebrate his birthday. His mother gives him 3 options to decorate the house with less cost. She said decorate the house either with balloon or flower or pictures. If child wants to decorate his house with balloons, then he has to inflates all the balloons one by one until 50 balloons are inflated with the help of the pump.

Real life steps as per the above the example:

1. Child seated with mother to discuss about decoration of house for celebrating his birthday
2. They discuss all 3 options to decorate the house with balloon or flower or pictures **(sequentially child think about every option with its cost)**

3. child think about all options and choose one option (he selects decoration of house with ballons) due to less cost. **Selection statement→use of [if/ elif /else]**

4. Child start inflating balloons one by one until 50 balloons are inflated with the help of pump. (Repetition of task; inflation of balloon -> **iteration statement → use of [for/ while]**

Note: student use this real-life example and make python program after learning python control statements

Selection Statements

- **if statement:**
  - Executes a block of code if a condition is true.
  - Example:

```
A=30
B=20
if (A>B):
  print(" A is greater than B")
```

- **if-else statement:**
  - Executes one block of code if a condition is true, and another block if it is false.
  - Example:

```
n=int(input('Enter a number'))
if n%2==0:
    print('EVEN number')
else:
    print('ODD number')
```

- **if-elif-else Statement:**
  - Checks multiple conditions in sequence, executing the first block of code where the condition is true.
  - Example:

```
n=int(input("Enter a no."))
if n>0:
    print("No. is Positive")
elif n<0:
    print("No. is Negative")
else:
    print("It is a zero")
```

Iterative Statements

While Loop:

- Repeats a block of code as long as a condition is true.

Example:

```
i=1
while i<=10:
    print(i,end=' ')
    i=i+1
```

For Loop

- Iterates over a sequence (e.g., list, tuple, string) or range of numbers.

Syntax: for <variable> in <sequence>
statements_ to_repeat

```
for i in range(10):
    print(i,end=' ')
```

- Example:

Jump statements

Python offers two types of jump statements to be used within loops to jump out of loop iterations.

- **break**
  Break statement is a jump statement which terminates the very loop it lies within and skips over a part of the code(i.e. rest of the loop) and jumps over to the statement following the loop.
- **continue**
  Continue is a jump statement which forces the next iteration of the loop to take place and skip the rest of the loop statements.

Looping with 'break' and 'continue':

-> break: Exit the loop when the '**break'** statement will encounter.
-> **continue**: Skip the rest of the code after '**continue'** and move to the next iteration.

- Examples:

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

```
for i in range(5):
    if i == 3:
        continue
    print(i)
```

→Example of **break**: In this case value will not be printed after 2 ,

→ Example of **continue**: in this case all the values will be printed as per the condition except 3

Same concept is applicable on '**while'** loop.

Loop with 'else' statement

The **else** statement of a Python loop executes when the loop terminates normally, i.e., when test condition results into false for a **while** loop or **for** loop has executed for the last value in the sequence; and not when the **break** statement terminates the loop.
Example:

Multiple Choice Questions

1.	What will be the output of the following code? print (type (3.0)) a) <class 'int'> b) <class 'float'> c) <class 'double'> d) <class 'decimal'>

2.	What is the primary purpose of the if statement in Python? a) To execute a block of code based on a condition b) To perform mathematical operations c) To repeat a block of code d) To define a function
3.	Ravi is a student of class xi computer sci and he is not able to understand the concept of **if** statement of Python and you as a programmer help him to find out the correct output of the following code snippet? **x = 10** **if x > 5:** **print("x is greater than 5")** a) x is greater than 5 b) x is less than 5 c) x is equal to 5 d) No output
4.	Which keyword is used to execute a block of code if the condition in the **if** statement is false? a) while b) elif c) else d) for
5.	What is the purpose of the **elif** statement in Python? a) To define a loop b) To perform arithmetic operations c) To exit from a loop d) To execute a block of code if the previous conditions are fals
6.	Sunita is a student of class xi computer sci and he is not able to understand the concept of **if-elif** statement of Python and you as a programmer help him to find out the correct output of the following code snippet? **x = 5** **if x < 3:** **print ("x is less than 3")** **elif x == 3:** **print ("x is equal to 3")** **else:** **print ("x is greater than 3")** a) x is less than 3 b) x is equal to 3 c) x is greater than 3 d) No output
7.	How can you execute multiple statements under a single if block in Python? a) Separate statements with a semicolon b) Indent the statements to the same level c) Use the and keyword between statements d) Use the elif keyword
8.	You as a programmer of python in a company, Find out the correct output of the following code **x = 10** **if x < 5:** **print ("x is less than 5")** **elif x > 15:** **print ("x is greater than 15")** **else:**

	print ("x is between 5 and 15") a) x is less than 5 b) x is greater than 15 c) x is between 5 and 15 d) No output
9.	What is the purpose of the while loop in Python? a) To execute a block of code repeatedly until a condition is false b) To execute a block of code a fixed number of times c) To define a function d) To iterate over items in a sequence
10.	What is the syntax of **while loop** in Python? a) while condition (): b) while condition: c) while condition loop: d) while loop condition:
11.	How can you exit a loop prematurely in Python? a) Using the break statement b) Using the continue statement c) Using the pass statement d) Using the exit function
12	Meera created a **List** named **fruits** in python and he **want to access each element of a List**. She writes below code but she is not sure whether below code is correct or not. Help her to find the correct output from the given options **fruits = ["apple", "banana", "cherry"]** **for fruit in fruits:** **print (fruit, end=' ')** a) No Output b) [“apple”, “banana”, “cherry”] c) 0 1 2 d) apple banana cherry
13.	What will the following code snippet print? **for i in range (2,-1,-1):** **print (i, end=' ')** a) 0 1 2 b) 1 2 3 c) 2 1 0 d) 3 2 1 0
14.	What is the purpose of the pass statement in Python? a) To exit from a loop prematurely b) To skip the current iteration of a loop c) To execute a block of code if a condition is false d) To do nothing and act as a placeholder
15.	What is the purpose of the else block in a loop in Python? a) To execute if the loop encounters an error b) To execute if the loop completes without encountering a break statement c) To execute if the loop encounters a continue statement d) To execute if the loop encounters a pass statement
16.	An empty/null statement in Python is a) go b) pass c) over d) ;
17.	What is the output of the following code? **x = 10** **if x > 5:** **print("Hello")** **elif x > 8:**

	print("Hi") **else:** **print("Hey")** a) Hey b) Hi c) Hello d) Hello, Hi

Answer (Multiple Choice Questions)

1	(b)<class 'float'>
2	(a) To execute a block of code based on a condition
3	(a) x is greater than 5
4	(c) else
5	(d)To execute a block of code if the previous conditions are false
6	(c) x is greater than 3
7	(b) Indent the statements to the same level
8	(c) x is between 5 and 15
9	(a) To execute a block of code repeatedly until a condition is false
10	(b) while condition:
11	(a) Using the break statement
12	(d) apple banana cherry
13	(c) 2 1 0
14	(d) To do nothing and act as a placeholder
15	(b) To execute if the loop completes without encountering a break statement
16	(b) pass
17	(c) Hello

Assertion and Reasoning Questions

Choose correct option for given Assertion (A) and Reasoning (R) a) Both A and R are true and R is the correct explanation of A. b) Both A and R are true but R is not the correct explanation of A. c) A is true but R is false. **d)** A is false but R is true.	
1	**Assertion (A):** The conditional flow of control can be defined with the help of if statement. **Reasoning (R)**: if statement executes one or more statements based on the given condition. If the condition evaluates to true, the statement block following the indentation gets executed, otherwise nothing gets executed.
2	**Assertion (A):** break and continue are termed as Jump statements.

	Reasoning (R): Jump statements can only be used with looping constructs but not with conditional constructs.
3	**Assertion (A**): In an if-else statement, the if block checks the true part whereas else checks for the false part. **Reasoning (R):** In a conditional construct, else block is mandatory.
4	**Assertion (A):** The data type of a variable is taken according to the type of value assigned to it. **Reasoning (R):** Data types do not require initialization at the time of declaration. This process is described as Dynamic Typing.

Answer (Assertion and Reasoning Questions)

1	a) Both A and R are true and R is the correct explanation of A.
2	a) Both A and R are true and R is the correct explanation of A.
3	c) A is true but R is false.
4	a) Both A and R are true and R is the correct explanation of A.

Short Questions with Answer

1	What will be the output of the following code? Also give short explanation about execution of loop with continue statement **for i in range (1, 6):** **if i == 3:** **continue** **print (i , end='')**
	Ans. 1 2 4 5 **Explanation:** The continue statement skips the rest of the loop and moves to the next iteration. So, when i equals 3, it skips printing that value.
2	What will be the output of the following code? with explanation, how the code will be executed? **for i in range(3):** **for j in range(3):** **print(i + j, end=' ')** **print()**
	Ans. 0 1 2 1 2 3 2 3 4 **Explanation:** The outer loop iterates three times, and for each iteration of the outer loop, the inner loop also iterates three times. The values of i and j are added together and printed. Each iteration of the outer loop starts a new line.
3	What will be printed by the following code? **num = 5** **while num > 0:** **print(num)** **num -= 1** **if num == 3:**

	break **else:** **print("Done")**
	Ans. 5 4
4	What is the output of the following code? **x = 10** **if x < 5:** **print("A")** **elif x > 7:** **print("B")** **else:** **print("C")**
	Ans. B
5	What will be the output of the following code? **x = 5** **while x > 0:** **print (x, end=" ")** **x - = 2** **if x == 1:** **break** **else:** **print("Done")**
	Ans. 5 3
6	What do you mean by continue and break keywords
	Ans. The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop. The continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues with the next iteration.
7	What is the use of While loop? With suitable example.
	Ans. The while loop in Python is used to repeatedly execute a block of code as long as a given condition remains true. **Example:** **c = 0** **while c < 5:** **print("C:", c)** **c += 1**
8	Find out the Error in the following Code Snippet. Rewrite the code after removing all the errors and underline.

```
x= int("Enter value of x:")
for in range [0,10]:
    if x=y
        print("They are equal")
    else:
        Print( "They are unequal")
```

Ans.

```
x= int(input("Enter value of x:"))
for y in range (0,10):
    if x==y:
        print("They are equal")
    else:
        print( "They are unequal")
```

Long Questions with Answer

1 Write a program to test whether given number is prime or not.

Ans:

```
a=int(input("Enter number: "))
k=0
for i in range(2,a//2+1):
    if(a%i==0):
        k=k+1
if(k<=0):
    print("Number is prime")
else:
    print("Number isn't prime")
```

2 Write a program to compute the result when two numbers and one arithmetic operator is given by user.

Ans:

```
a = int(input('Enter 1st number: '))
b = int(input('Enter 2nd number: '))
c = input('Enter the Operator +,-,/,*: ')
print("The result is: ",end='')
if c=='+':
    print(a+b)
elif c=='-':
    print(a-b)
elif c=='/':
    print(a/b)
elif c=='*':
    print(a*b)
else:
    print('Error : Wrong operator')
```

4 Write a program to find the sum of first n natural numbers

Ans:

```
n=int(input("Enter the Limit : "))
s=0
for i in range(1,n+1):
    s=s+i
print("The sum is : ",s)
```

3 Write a program to input a digit from 0 to 9 and print it in words.

Ans:

```
n=int(input("Enter the Digit from 0 to 9: "))
print("Entered Digit is : ",end='')
if n==0:
    print("Zero")
elif n==1:
    print("One")
elif n==2:
    print("Two")
elif n==3:
    print("Three")
elif n==4:
    print("Four")
elif n==5:
    print("Five")
elif n==6:
    print("Six")
elif n==7:
    print("Seven")
elif n==8:
    print("Eight")
elif n==9:
    print("Nine")
else:
    print("Not a digit")
```

5 Write a program to find the sum of first n odd numbers.

Ans:

```
n=int(input("Enter the Limit : "))
s=0
for i in range(1,n+1,2):
    s=s+i
print("The sum is : ",s)
```

6 Write a program to print the following pattern

(a) Input value **is 5**	(b) input sting is **'INDIA'**	(c)print below	(d) print below
*	I	A	*****
**	IN	BB	****
***	IND	CCC	***
****	INDI	DDDD	**
*****	INDIA	EEEEE	*

(a)

```
n=int(input("Enter the Limit : "))
for i in range(1,n+1):
    for j in range(1,i+1):
        print("*",end='')
    print("")
```

(b)

```
s=input("enter a string")
n=len(s)
for i in range(0,n):
  for j in range(0,i+1):
    print(s[j],end='')
  print('')
```

Ans:

(c)

```
num=65
for i in range (0, 5):
  for j in range (0, i+1):
    ch = chr (num)
    print (ch, end=' ')
  num = num + 1
  print (" ")
```

(d)

```
n=5
for i in range (n, 0, -1):
    for j in range (1, i + 1):
        print ('*', end='')
    print (" ")
```

7 Write a program to calculate the roots of a given quadratic equation. If quotients

	will be entered by the user.

	Ans:
	`import math`
	`a=int(input("Enter a "))`
	`b=int(input("Enter b "))`
	`c=int(input("Enter c "))`
	`d=(b*b)-(4*a*c)`
	`if d>=0:`
	`  print("roots are : ")`
	`  x1=-b+math.sqrt(d)/(2*a)`
	`  x2=-b-math.sqrt(d)/(2*a)`
	`  print(" x1 = =",x1)`
	`  print(" x2 = =",x2)`
	`else:`
	`  print("roots are imaginary.")`

Case Based Questions with Answer

1	Krishna is looking for his dream job but has some restrictions. He loves Delhi and would take a job there if he is paid over Rs.40,000 a month. He hates Chennai and demands at least Rs. 1,00,000 to work there. In any another location he is willing to work for Rs. 60,000 a month. The following code shows his basic strategy for evaluating a job offer.
	`pay= _____`
	`location= _____`
	`if location == "Mumbai":`
	`  print ("I'll take it!") #Statement 1`
	`elif location == "Chennai":`
	`  if pay < 100000:`
	`  print ("No way") #Statement 2`
	`  else:`
	`  print("I am willing!") #Statement 3`
	`elif location == "Delhi" and pay > 40000:`
	`  print("I am happy to join") #Statement 4`
	`elif pay > 60000:`
	`  print("I accept the offer") #Statement 5`
	`else:`
	`  print("No thanks, I can find something better")#Statement 6`
	On the basis of the above code, choose the right statement which will be executed when different inputs for pay and location are given.

	(A) **Input value:** location = "Chennai", pay = 50000 a) Statement 1 b) Statement 2 c) Statement 3 d) Statement 4 **Answer: (b) Statement 2**
	(B) **Input value:** location = "Surat", pay = 50000 a) Statement 2 b) Statement 4 c) Statement 5 d) Statement 6 **Answer: (d) Statement 6**
	(C) **Input value:** location = "Any Other City", pay = 1 a) Statement 1 b) Statement 2 c) Statement 6 d) Statement 4 **Answer: (c) Statement 6**

	(D) **Input value:** location = "Delhi", pay = 500000 a) Statement 6 b) Statement 5 c) Statement 4 d) Statement 3 **Answer: (c) Statement 4**
	(E) **Input value:** location = "Lucknow", pay = 65000 a) Statement 2 b) Statement 3 c) Statement 4 d) Statement 5 **Answer: (d) Statement 5**
2	Sanjeev is, a Python Programmer he is trying to write a program of finding leap year if the value of year is given by user (A year is a leap year if it is divisible by 4, except that years divisible by 100 are not leap years unless they are also divisible by 400.)

	Ans:
	`year = int (input ("Enter year: "))`
	`if year % 400 == 0:`
	`  print (year, "is a Leap Year")`
	`elif year % 100 == 0:`
	`  print (year, "is not a Leap Year")`
	`elif year % 4 == 0:`
	`  print (year, "is a Leap Year")`
	`else:`
	`  print (year, "is not a Leap Year")`
	OUTPUT:
	Enter year: 1800
	1800 is not a Leap Year

3	Read the following case and answer the questions:
	A school is organizing a mock election. The eligibility to vote is based on the age of the student. The program checks the student's age and prints if the student is eligible to vote or not.
	`age = int(input("Enter your age: "))`
	`if age >= 18:`
	`  print("Eligible to vote.")`
	`else:`
	`  print("Not eligible to vote.")`
	Questions:
	a) What will be the output if the user inputs 20?
	b) What is the purpose of the int() function in this code?
	c) Modify the code to also print "Please wait until you are 18." if not eligible.
	d) What will happen if the user enters a non-numeric input?
	e) If a student is eligible to vote another message must be displayed on the screen that “choose a right person” , after which print statement, would you add this line of code.

	Answer: **(a)** Eligible to vote. a) It converts the input string to an integer.

b)

```
age = int(input("Enter your age: "))
if age >= 18:
    print("Eligible to vote.")
else:
    print("Not eligible to vote.")
    print("Please wait until you are 18.")
```

c) It will raise a ValueError since int() cannot convert non-numeric input.

d) "choose a right person" line of code will be added in **if block statement**

4

A school uses the following grading system:

Marks >= 90 → Grade A

Marks >= 75 and < 90 → Grade B

Marks >= 50 and < 75 → Grade C

Marks < 50 → Grade D

```
marks = int(input("Enter your marks: "))

if marks >= 90:
    print("Grade A")
elif marks >= 75:
    print("Grade B")
elif marks >= 50:
    print("Grade C")
else:
    print("Grade D")
```

Questions:

a) What grade will be displayed for 88 marks?
b) What is the output if the marks entered are 45?
c) Modify the code to also check if the marks are within 0 to 100.
d) What will happen if the user enters -10 in the above mentioned code.

Answers:

a) Grade B

b) Grade D

c)

d) The original code will still run and give **Grade D**, but it should be considered

```
marks = int(input("Enter your marks: "))
if marks < 0 or marks > 100:
    print("Invalid marks.")
elif marks >= 90:
    print("Grade A")
elif marks >= 75:
    print("Grade B")
elif marks >= 50:
    print("Grade C")
else:
    print("Grade D")
```

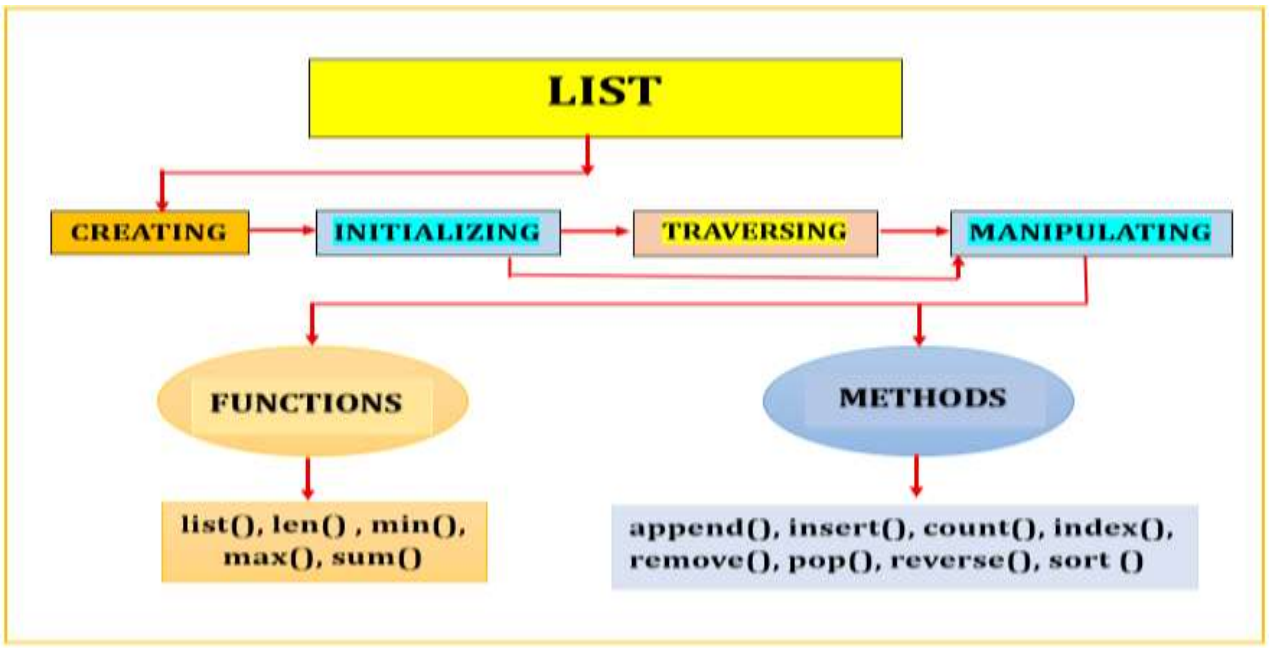
invalid input.

5	A school wants to help students in revise multiplication tables. The following code print the multiplication table of a number:
	Hint- range*(start, stop, step)*)
	start position: default 0 if not specify**, stop position:** at which position to stop,
	step position: increment /decrement, its optional, default **is 1**
	`num = int (input ("Enter a number: "))`
	`for i in range (1, 11):`
	`  print (num, "x", i, "=", num * i)`
	Questions:
	a) What will be the output for input 5?
	b) How many times will the loop run?
	c) Modify the code to print the table in reverse order (from 10 to 1).
	d) What is the role of range (1, 11)?

	Answers: a) 5 x 1 = 5 5 x 2 = 10 … 5 x 10 = 50 b) 10 times. c) for i in range (10, 0, -1): print (num, "x", i, "=", num * i) d) It generates numbers from 1 to 10.

List in Python

Mind Map:



List is an ordered sequence which is mutable and made up of one or more elements. A list can have elements of different data types such as integer, float, string, tuple or even another list.

Elements of a list are enclosed in square brackets and are separated by comma. It is a mutable data type.

Creating List

- We can create a list by placing elements inside square brackets [], separated by commas.
  Example : Number=['One', 'Two', 'Three']
- We can create list using list() method
  data=list() Creates empty list
  data=list([3,4,5])

Accessing Elements in a List: Each element in the list is accessed using a value called index. The first index value is 0 and second index value is 1 and so on, if its move from left to write.
If accessing of element start from right to left then the index value start from -1 to -n

Traversing a List: We can access each element of the list or traverse a list using a loop.
Example:

```
L=[1,2,3,4]
for item in L:
    print(item)
```

Nested List: When a list appears as elements of another list, it is called a nested list.
Example: L=[1,2,[3,4],5]

List Methods and Built-in Functions

There is a key difference between functions and methods in Python. Functions take objects as inputs/parameter. Methods in contrast act on objects.

Function	**Method**
A= [25,36,37,12,25] M=len(A) print(M) #output **5** **list named A passed as parameter**	A= [25,36,37,12,25] M=A.count(25) print(M) #output **2** **count() method acts on object A (list)**

Function	Syntax	Description
list()	list()	If no argument is passed, it will create an empty list.
list()	list(sequence)	It returns a list created from the passed arguments, which should be a sequence type (string, list, tuple etc.). if no argument is passed, it will create an empty list.
len()	len (list)	Returns the length of the list i.e. number of elements in a list
max()	max(list)	Returns the element with the maximum value from the list
min()	min(list)	Returns the element with the minimum value from the list
sum()	sum(list)	It returns sum of elements of the list.
sorted ()	sorted (sequence, reverse=False)	It returns a newly created sorted list; it does not change the passed sequence. By default in sorted() reverse is False

Method	Syntax	Description
append()	list.append (items)	It adds a single item to the end of the list.
insert()	list.insert (index_no, value)	It adds an element at a specified index
reverse()	list.reverse ()	It reverses the order of the elements in a list
index()	list.index (item)	It returns the index of first matched item from the list.
sort()	list.sort ()	This function sorts the items of the list
count()	list.count (element)	It counts how many times an element has occurred in a list and returns it.
pop()	list.pop (index)/ list.pop()	It removes the element from the specified index and also returns the element which was removed. Default index is -1.
remove()	list.remove (value)	It is used when we know the element to be deleted, not the index of the element.
clear()	list.clear ()	.It removes all the elements from the list.

List Methods and Built-in Functions with Examples

The values that make a list are called its elements, or its items. We will use the term **element** or **item** means same.

Functions	Examples with output
list()	X=list() print(X) **Output:** [ ] #emply list created
list(sequence)	Lst=list ("hi") print (Lst) **Output:** **['h', 'i']** **#sequence(String) converted in to list**
len (list)	L= [2, 3, [4,5]] print(len(L)) **Output:** **3** # Explanation: In above example 3rd element is list and it is considered as single element
max(list)	L=[1,-2,4,-4] print(max(L)) **output: 4** **Try it!!!** lst=[[2,3],[4,6],[6,4]] print(max(lst))
min(list)	L=[1,-2,4, -4] print(min(L)) **output: - 4**
sum(list)	L=[1, -2,4, -4] print(sum(L)) **output**: -1

sorted(list,reverse=False)	L=[15,16,17,2,3,9] print(sorted(L)) #by default reverse False **output:** **[2,3,9,15,16,17]** --- L=[15,16,17,2,3,9] print (sorted (L, reverse=True)) **output:** **[17,16,15,9,3,2]**
Methods	**Examples with output**
list.append(items/ elements)	lst=[3,6,0] lst.append(9) print(lst) **output :** **[3,6,0,9]**
list.insert (index_no, value)	Odd=[1,5,7,9] Odd.insert(1,3) print(Odd) **Output:** **[1,3,5,7,9]** **# value 3 inserted at index position 1**
list.reverse ()	Even=[2,4,6,8] Even.reverse() print(Even) **output:** **[8,6,4,2]**
list.index (item/element)	Even=[2,4,6,8] print(Even.index(4)) **Output:** **1** #element 4 at index position 1
list.sort ()	Num=[2,4,-6,-8,0] Num.sort() **#by default reverse is False** print(Num) **Output:** **[-8, -6, 0, 2, 4]**
	Num= [2,4, -6, -8,0] Num.sort(reverse= True) print (Num) **Output:** **[4, 2, 0, -6, -8]** Use the “reverse” argument to sort in descending order

list.count (element)	data= [10,30,20,30,90,30] print(data.count(200)) **output:** **0** # element /value 200 not in the list data
	data=[10,30,20,30,90,30] print(data.count(30)) **Output:** **3** **#** element /value 30 three times in the list data

list.pop (index)	data=[10,30,20,40,50]
	print(data.pop()) # pop() without index position
	print(data)
	Output:
	50 # pop () returns the popped value
	[10,30,20,40]
	# data list printed after popped its last element

	data= [10,30,20,40,50]
	print(data.pop(2)) #pop() function with index position
	print(data)
	Output:
	20 # pop () returns the popped value
	[10,30,40,50]
	# data list printed after popped its element of particular index position.

list.remove (item/element)	data=[10,30,20,25,30,40]
	data.remove(30) #element **30** at index position 1, 4
	print(data)
	output:
	[10, 20, 25, 30, 40]
	# remove the first occurrence of a specified value from the list

list.clear ()	data=[10,30] data.clear() print(data) **output:** **[ ]** # empty list will be displayed after implanting clear method

Multiple Choice Questions

1.	Suppose list1 is [3, 4, 5, 20, 5, 25, 1, 3], what is list1 after execution of list1.pop (1)? a) [3, 4, 5, 20, 5, 25, 1, 3] b) [1, 3, 3, 4, 5, 5, 20, 25] c) [3, 5, 20, 5, 25, 1, 3] d) [1, 3, 4, 5, 20, 5, 25]

2.	Suppose a list mylist is [10,23,45,3,34,50], what is mylist after execution of mylist.remove(3) a) [10, 23, 3, 34, 50] b) [10, 23, 45, 34, 50] c) [10, 23, 3, 34, 50] d) No output
3.	Consider the following code and predict the output mylist = [2,3,8,9,3] mylist.append([2,4]) print(mylist) a) [2, 3, 8, 9, 3, 2, 4, [2, 4]] b) [[2, 4], 2, 3, 8, 9, 3, 2, 4] c) [2, 3, 8, 9, 3, ,2, 4, 2, 4] d) [2, 3, 8, 9, 3, [2, 4]]
4.	Consider a list mylist with n elements, in mylist.pop() what is the default index a) 1 b) 0 c) -1 d) 4
5.	What is the output when we execute? print(list("hello")) a) ['h', 'e', 'l', 'l', 'o'] b) ['hello'] c) will not create a list d) ['h,e,l,l,o']
6.	Which of the following commands will create a list? a) listl = list( ) b) listl = [ ] c) listl = list([1, 2, 3]) d) all of these

Answer (Multiple Choice Questions)

1.	c) [3, 5, 20, 5, 25, 1, 3]
2.	b) [10, 23, 45, 34, 50]
3.	d) [2, 3, 8, 9, 3, [2, 4]]
4.	c) -1
5.	a) ['h', 'e', 'l', 'l', 'o']
6.	d) all of these

Assertion and Reasoning Questions

Choose correct option for given Assertion (A) and Reasoning (R)
a) Both A and R are true, and R is the correct explanation of A.
b) Both A and R are true, but R is not the correct explanation of A.
c) A is true, but R is false.
d) A is false, but R is true.

1.	**Assertion (A):** Lists in Python are mutable. **Reason (R):** We can modify, add, or remove elements from a list after its creation.
2.	**Assertion (A):** A list in Python can contain another list as its element. **Reason (R):** Python does not allow different data types inside a list.

Answer(Assertion and Reasoning Questions)

1	**Answer:** a) Both A and R are true, and R is the correct explanation of A.

2	**Answer:** c) A is true, but R is false.

Short Questions with Answer

1. Write a python code to read a list of numbers and print all even numbers from the list.

Answer:

```
xlist=eval(input("enter elements"))
for i in xlist:
   if i%2==0:
      print(i)
```

Output :

```
enter elements [1,2,3,8,9]
2
8
```

2. Write a python code to read a list and replace all even elements with 0 and odd with 1

Answer:

```
xlist=eval(input("enter elements"))
for i in range(len(xlist)):
   if xlist[i]%2==0:
      xlist[i]=0
   else:
      xlist[i]=1
print(xlist)
output:
enter elements[2,5,7,98,0,7]
[0, 1, 1, 0, 0, 1]
```

3. Predict the output of the following code: -

```
Moves=[11, 22, 33, 44]
Queen=Moves
Moves[2]+=22
L=len(Moves)
for i in range (L):
      print ("Now@", Queen[L-i-1], "#", Moves [i])
```

Answer:

```
Now@ 44 # 11
Now@ 55 # 22
Now@ 22 # 55
Now@ 11 # 44
```

4. Consider the following list **myList**. What will be the output of the code after executing of below code:

```
myList = [10,20,30,40]
myList.append([50,60])
d=myList.pop(2)
```

```
print(d)
print(myList)
```

Answer:

```
30
[10, 20, 40, [50, 60]]
```

Unsolved Questions (Test your understanding)

1. Read a list and display all elements ends with 5.
2. Read a list and count the occurrence of 5 in the list.
3. Read list and remove last element.
4. Read a list and print all positive numbers from the list
5. Read a list and display elements along with index
6. Write python code to read a list lst and copy all the negative numbers from the list lst to another list lst_neg
7. Read a list and add 2 to all even elements in the list and display the updated list.
8. Write the output of the following code:

```
xylist=[2,3,5,4,56,0,-1]
xylist.pop()
xylist.pop(-1)
xylist.insert(2,4)
print(len(xylist))
print(xylist)
```

9. What will be the output of the following code:

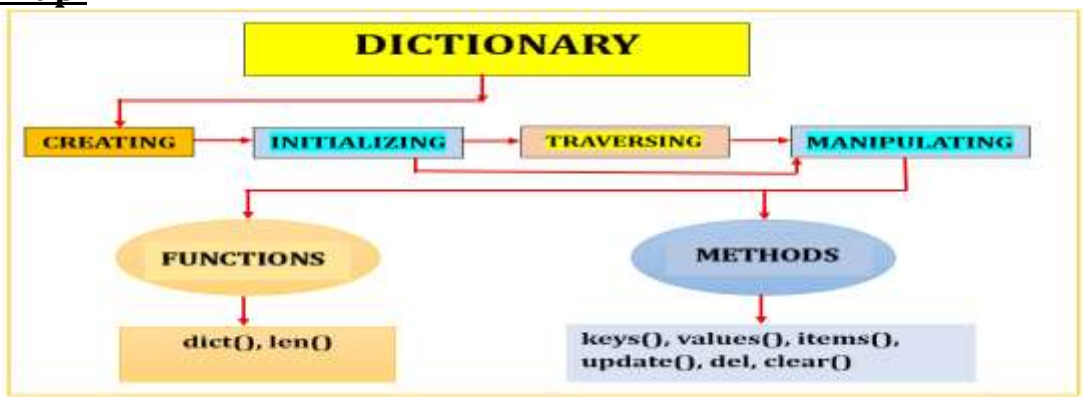
```
list1 = [12,32,65,26,80,10]
list1.sort()
print(list1)
```

10. What will be the output of following code

```
myList = [1,2,3,4,5,6,7,8,9,10]
newlist=[]
for i in range(len(myList)):
  if i%2 == 0:
     newlist.insert(i,myList[i])
print(newlist)
```

Dictionary in Python

Mind Map:



The data type dictionary falls under mapping. It is a mapping between a set of **keys** and a set of **values**. The key-value pair is called an item. A key is separated from its value by a colon (:) and consecutive items are separated by commas. Items in dictionaries are unordered,
A dictionary is an unordered sequence of key-value pairs.

Key and value in a **key-value** pair in a dictionary are separated by a colon. Further, the key: value pairs in a dictionary are separated by commas and are enclosed between curly parentheses.

- The keys of the dictionaries are immutable types such as Integers or Strings etc.
- Indices in a dictionary can be of any **immutable type** and are called **keys**.
- The values of a dictionary in Python are mutable.

Creating Dictionaries

A Dictionary can be created in three different ways:

1. Empty Dictionary using empty { }

```
D = { } # Empty Dictionary
```

2. Dictionary using literal notation

```
D = {"Name": "Mohan", "Class": "XI", "City": "Gurdaspur"}
print(D)
```

3. Dictionary using dict() function

Dict() function is used to create a new dictionary with no items. For example,

```
Months = dict() # Creates an empty dictionary
print(Month) # Prints an empty dictionary
```

We can use Square Brackets **[ ]** with keys for accessing and initializing dictionary values.
For Example:

```
Months[0] = 'January'
Months[1] = 'February'
Months[2] = 'March'
print(Months)
```

output: {0 : 'January', 1: 'February' , 2 : 'March'}

```
Months = dict(Jan = 31, Feb = 28, March = 31)
# Creating dictionary by giving values in dict() function
print(Months)
```

output: {'Jan': 31, 'Feb': 28, 'March': 31}

Accessing Elements of a Dictionary

Elements of Dictionary may be accessed by writing the Dictionary name and key within square brackets ([ ]) as given below:

```
D = {0 : "Sunday", 1 : "Monday", 2: "Tuesday"}
print(D[1])
```

output: Monday

***Attempting to access a key that does not exist, causes an error.

Traversing a Dictionary:

Dictionary items can be accessed using a for loop.

```
d= {"A": "Apple","B": "Boy", "C":"Cat"}
for i in d:
   print(d[i])
```

output:

Apple

Boy

Cat

Adding an Element in a Dictionary

We can add new element (key : value pair) to a dictionary using assignment, but the key being added must not exist in dictionary and must be unique.

```
d["D"]="Dog"    # if a new key is given, new item is added
print(d)
```

output: {'A': 'Apple', 'B': 'Boy', 'C': 'Cat', **'D': 'Dog'**}

A new key **'D'** added in to the dictionary **d**

Updating / Modifying an element in Dictionary

We can change the individual element of dictionary as given below:

```
d["A"]="Android"    # Value of Key ("A") is changed
print(d)
```

output: {'A': **'Android** ', 'B': 'Boy', 'C': 'Cat', 'D': 'Dog'}

Dictionary Methods and Built-in Functions

Function	Syntax	Description
dict()	dict()	Creates a dictionary from a sequence of key-value pairs
len()	len (dictionary)	function: It is used to find the length of the dictionary, i.e., the count of the key : value pair.
keys()	dictionary.keys()	Returns a list of keys in the dictionary
values()	dictionary.values()	Returns a list of values in the dictionary
items()	dictionary.items()	Returns a list of tuples(key – value) pair
update()	dictionary.update(iterable)	appends the key-value pair of the dictionary passed as the argument to the key-value pair of

		the given dictionary
del	del Dict_name del dictionary[key]	Deletes the item with the given key or to delete the dictionary from the memory.
clear()	dictionary.clear()	Deletes or clear all the items of the dictionary

Dictionary functions with examples:

Function	**Examples**	**output**
dict()	d=dict() print(d)	{}
	dates=dict([['Jan',31], ['Feb',28], ['March',31]]) print(dates)	{'Jan': 31, 'Feb': 28, 'March': 31}
len (dictionary)	dates={'Jan': 31, 'Feb': 28, 'March': 31} print(len(dates))	3
Method	**Examples**	**output**
dictionary.keys()	dates={'Jan': 31, 'Feb': 28, 'March': 31} print(dates.keys())	dict_keys(['Jan', 'Feb', 'March']) **Returns a list containing the dictionary's keys**
dictionary.values()	dates={'Jan': 31, 'Feb': 28, 'March': 31} print(dates.values())	dict_values([31, 28, 31]) **Returns a list of all the values in the dictionary**
dictionary.items()	dates={'Jan': 31, 'Feb': 28, 'March': 31} print(dates.items())	dict_items([('Jan', 31), ('Feb', 28), ('March', 31)]) **returns a list containing a tuple for each key value pair**
dictionary.update (iterable)	dates={'Jan': 31, 'Feb': 28, 'March': 31} dates.update({'Feb':29}) print(dates)	{'Jan': 31, 'Feb': **29,** 'March': 31}
	dates={'Jan': 31, 'Feb': 28, 'March': 31} dates.update({'April':30}) print(dates)	{'Jan': 31, 'Feb': 28, 'March': 31, 'April': 30} **# 'April' Key not exist in dates so new key 'April' added in to the dictionary**
del Dict_name	d={1:"One",2:"Two"} del d print(d)	Removes entire dictionary **#print statement will cause an error because 'd' dictionary no longer exists.**
del dictionary[key]	dates= {'Jan': 31, 'Feb': 28, 'March': 31} print ("Before removal") print(dates) del **dates["March"]** print ("After removal")	Before removal {'Jan': 31, 'Feb': 28, 'March': 31} After removal {'Jan': 31, 'Feb': 28} **#will delete particular key from**

	print(dates)	**a dictionary**
dictionary. clear()	d={"A":"An","Is":"Are"} d.clear() print(d)	{ } **Removes all the elements from the dictionary**

Note: if single **key:value** pair is to remove from a dictionary then **pop()** and **popitem()** method of dictionary can be used

Checking existence of a key in a dictionary

To check if a key is present in the dictionary or not we can use the **in** operator – if the given key is present in the dictionary, it returns True, otherwise it returns False.
e.g.: Consider the dictionary:

```
dates= {'Jan': 31, 'Feb': 28, 'March': 31}

'Feb' in dates        # returns True
'FEB' in dates        # returns False
'FEB' not in dates    # returns True
```

Multiple Choice Questions

1	Dictionaries are..... set of elements. a) sorted b) Ordered c) unordered d) random
2	What would the following code print? d= {"Banana": "Yellow", "Orange": "Orange", "Papaya": "Orange"} print(d["Orange"]) a) Yellow b) Orange c) Papaya d) Banana
3	What would the following code print? dates={'Jan': 31, 'Feb': 28, 'March': 31, 'Feb':29} print(len(dates)) a) 1 b) 4 c) 3 d) 0
4	dates={'Jan': 31, 'Feb': 28, 'March': 31, 'feb':29} print(len(dates)) a) 1 b) 4 c) 3 d) 0
5	Which of the following statements create a dictionary? i) d = {} ii) d1 = dict([["john",40],["peter",45]]) iii) d2 =dict(1,2,4,5) a) Only i b) Only ii c) Both i and ii d) all (i, ii, iii)
6	Which of these about a dictionary is false?

	a) The values of a dictionary can be accessed using keys b) The keys of a dictionary can be accessed using values c) Dictionaries store data as key : value pair d) Dictionaries are mutable
7	Given the following dictionary Day={1:"Monday", 2: "Tuesday", 3: "Wednesday"} Which statement will be used to delete a value 'Tuesday' a) del Day b) del Day[1] c) del Day[2] d) del Day['Tuesday']
8	Identify the invalid Python statement from the following: a) d = dict() c) f =dict([('Name', 'Sahil')]) b) e = {} d) g= dict{}
9	State True or False : "In Python, Dictionary is a mutable data type".

Answer (Multiple Choice Questions)

1	c) unordered
2	b) Orange
3	c) 3
4	b)4
5	c) Both i and ii
6	b) The keys of a dictionary can be accessed using values
7	b) del Day[2]
8	d) g=dict{}
9	True

Assertion and Reasoning Questions

Choose correct option for given Assertion (A) and Reasoning (R)
a) Both A and R are true, and R is the correct explanation of A.
b) Both A and R are true, but R is not the correct explanation of A.
c) A is true, but R is false.
d) A is false, but R is true.

1.	**Assertion (A):** Dictionaries in Python are mutable. **Reason (R):** Items in a dictionary can be added, removed, or updated after creation.
2.	**Assertion (A):** Dictionaries are enclosed within curly braces { }. **Reason (R):** The key-value pairs are separated by commas (,).

Answer(Assertion and Reasoning Questions)

1	a) Both A and R are true, and R is the correct explanation of A
2	b) Both A and R are true but R is not the correct explanation of A

Short Questions with Answer

1.	Rita is a student of class XI. Help her to create a dictionary to store details of 10 employees given by the user. Each element consists of empcode as key, name and salary as values.
	
	Answer:
	emp={}
	for i in range(10):
	  l=[]
	  empcode=int(input("enter emplyee code"))
	  name=input("enter name")
	  salary=int(input("enter salary"))
	  l.append(name)
	  l.append(salary)
	  emp[empcode]=l
	print(emp)

2.	Consider the following dictionary and print the name and salary of employees in "computer" department.
	{ "anil":["computer",10000],
	"anju":["History",5000],
	"somu":["computer",11000],
	"sam":["English",8000] }
	
	Answer:
	data={"anil":["computer",10000], "anju":["History",5000],
	"somu":["computer",11000], "sam":["English",8000] }
	for key, value in data.items():
	  if value[0]=="computer":
	  print(key , value[1])

Unsolved Questions (Test your understanding)	
1.	The dictionary empdat contains employee records with the following values {1 :["Raj " ,85000], 2: ["Sathya ", 20000], 3: ["Meenu ",89000]} Write a Python program to display only the records where the employee's salary exceeds 75000.
2.	The dictionary students stores students name and their marks:

	students = { "A101": ["Ravi", 88], "A102": ["Divya", 76], "A103": ["Kiran", 95], "A104": ["Neha", 65]} Write a Python program to display the name and marks of students who scored more than 80
3.	Write python code to create dictionary to store the following details of 10 products {Productno: [Product name, price, quantity]}
4.	Write the Python statement for the following tasks using built-in functions/methods only: To remove the item whose key is "NISHA" from a dictionary named Students. For example, if the dictionary Students contains { "ANITA":90, "NISHA":76, "ASHA":92) After removal, the dictionary should contain {"ANITA":90, "ASHA":92}

Introduction to NumPy

Introduction, Creation of NumPy Arrays from List

NumPy ("Numerical Python" or Numeric Python") is a package / module for data analysis, specially used for scientific analysis of data with Python. NumPy provides different functions for fast mathematical computation on arrays, matrices and on multi-dimensional arrays.
To make array using NumPy in Python, need to install a NumPy. NumPy is an **open-source python library** that can be installed using Python packages like pip.

To install NumPy in Python, use the pip package on command line and choose the absolute path of location, where python is installed.

pip install numpy

NumPy is a library in Python which can be imported in a program by using import command as

import numpy as np

Array: Group or collection of similar type of elements.
Eg. Scores of players in a cricket match, Marks of students in a class

NumPy Array: A grid contains value of same type (homogeneous elements) and uses Zero based indexing, like lists in Python. NumPy Array also known as **ndarray**

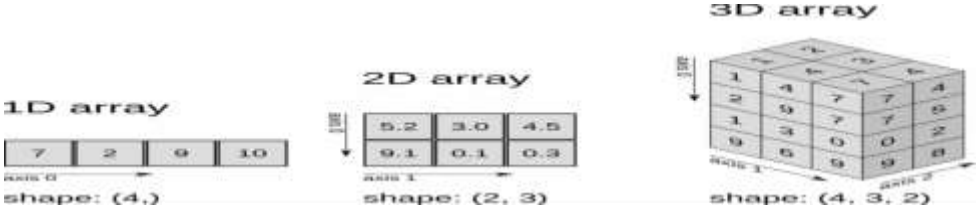
Difference between List and Array:

- Arrays only hold data of same data type, while list can hold the data of mixed data types (like int, float etc)
- Using of Arrays to store data is more memory efficient and help to perform fast calculation and operations.
- List is a part of core python , Array/ndarray is a part of NumPy Library

Creation of NumPy Arrays from List:

NumPy Arrays are grid-like structures similar to lists in Python but optimized for numerical operations. NumPy array can be created by converting a regular Python list into an array using the **np.array()** function. This function returns **ndarray** object. **Once the NumPy array is defined, the space it occupies in memory is fixed and cannot be change.**

Ndarray is one of the most important classes in the NumPy python library. It is basically a multidimensional or n-dimensional array of fixed size with homogeneous elements (i.e., the data type of all the elements in the array is the same). A multidimensional array looks something like this:



Axes: Axes are defined for arrays with more than one dimension. A 2-dimensional array has two corresponding axes: the first running vertically downwards across **rows(axis 0),** and the second running horizontally across **columns (axis 1).** Many operations can take place along one of these axes.

Rank: The no. of axes in a **ndarray** is called its rank.

Creation of NumPy array using List

Example: Python Code

```
import numpy as np
Lvalue=[10,20,30,40,50]
Larray=np.array(Lvalue)
```

Types of Arrays:

- 1-D Array: A single row of elements.
- 2-D Array: A matrix with rows and columns

1) 1D NumPy array creation -> Using List:

```
>>> import numpy as np
>>> List1=[10,50,90,130]
>>> Arr1=np.array(List1)
>>> Arr1
array([ 10,  50,  90, 130])
>>> print(List1)
[10, 50, 90, 130]
>>> print(Arr1)
[ 10  50  90 130]
```

2) 2D NumPy Array creation -> Using list:

A list of lists i.e. nested list will create a 2D Numpy array, similarly, you can also create N-dimensional arrays.

```
>>> import numpy as np
>>> List2=[[1,2,3],[4,5,6]]
>>> Arr2=np.array(List2)
>>> Arr2
array([[1, 2, 3],
       [4, 5, 6]])
>>> print(List2)
[[1, 2, 3], [4, 5, 6]]
>>> print(Arr2)
[[1 2 3]
 [4 5 6]]
```

Attributes of Numpy array:

SNo.	Attribute	Syntax	Example
1	**Shape**	**<ndarrayname>.shape** A tuple of integers giving the size of the array along each dimension. (no. of element along each axis of the array)	import numpy as np arr = np.array([[1, 2, 3, 4, 5]]) print(arr.shape) **(1,5) # output** import numpy as np arr = np.array([[1, 2, 3, 4, 5], [2,3,4,5,6]]) print(arr. shape) **(2,5) # output**

2	**Size**	**<ndarrayname>.size**
		Total no. of elements in the array
			>>>Arr1=np.array([1,2,3])
			>>> Arr1.size
			3 #output
			>>>Arr2=np.array([[1,2],[3,4],[5,6]])
			>>> Arr2.size
			6 #output

3	**Datatype (dtype):**	**<ndarrayname>.dtype**
		Type of data stored in the array
			>>> Arr3=np.array([1,20,60])
			>>> Arr3.dtype
			int32 / int64
			>>> Arr4=np.array([10.5,25.8,50.7])
			>>> Arr4.dtype
			float64

4	**ItemSize:**	**<ndarrayname>.itemsize**
		Size of each element of ndarray in bytes
			>>> arr1=np.array([10,20])
			>>> arr1.itemsize
			4 (As all elements are int type)
			>>> arr2=np.array([2.5,7.8])
			>>> arr2.itemsize
			8 (As all elements are float type)

5	**ndim**	**<ndarrayname>.ndim**
		It returns an integer representing the number of dimensions.
			import numpy as np
			Arr2=np.array([1,2,3])
			print(Arr2.ndim)
			1 #output
			Arr2=np.array([[1,2,3], [7,8,9]])
			print(Arr2.ndim)
			2 #output

Creating arrays with specific values:

method	Description	Example

numpy.zeros()	This function creates an array filled with zeros. It requires the shape of the array as a parameter.	>>> a1=np.zeros((2,3))
		>>> a1
		array([[0., 0., 0.],
		[0., 0., 0.]])
		>>> a2=np.zeros((4,))
		>>> a2
		array([0., 0., 0., 0.])

numpy.ones()	This function creates an array filled with ones. It requires the shape of the array as a parameter.	>>> a3=np.ones((2,3))
		>>> a3
		array([[1., 1., 1.],
		[1., 1., 1.]])
		>>> a4=np.ones((2,))
		>>> a4
		array([1., 1.])

numpy.arange()	It creates an array of evenly spaced values within a given interval. It is similar to Python's built-in range() function but returns a NumPy array instead of a list.
	numpy.arange([start,]stop, [step,] dtype = None)
		>>> ar=np.arange(5,10)
		>>> ar
		array([5, 6, 7, 8, 9])
		>>> ar2=np.arange(5)
		>>> ar2
		array([0, 1, 2, 3, 4])
		>>>ar3=np.arange(5.5, 10.5,2,dtype='float64')
		>>> ar3
		array([5.5, 7.5, 9.5])

numpy.full()	Return a new array with the same shape and type as a given array filled with a fill_value. Parameters	>>>ar1=np.full((2,2),5,dtype='int') >>> ar1 array([[5, 5], [5, 5]])

	numpy.full(shape, fill_value, dtype = None)	>>>ar2=np.full((4,),1.5, dtype='float') >>> ar2 array([1.5, 1.5, 1.5, 1.5])

Multiple Choice Question and Fill in the Blanks

1	What does NumPy stand for? (a) Number Platform (b) Numerical Python (c) Numeric Python (d) Number Picture
2	Array object in NumPy called _____ (a) number array (b) Narray (c)numpy Array (d)ndarray
3	NumPy is open source in python (true/ false)
4	Follow the below code : arr = np.array([[1, 2, 3,4], [4, 5, 6,7]]) Find out the dimensions it has. (a) 1 (b) 2 (c) 3 (d) 4
5	Fill the blanks in the below python code and underline it. import numpy as _____ Array=np._____([1,2,3,5,6]) print(Array)
6	Which attribute of NumPy array is used to find out the number of dimensions? (a) ndimn (b) ndim (c) ndimensions (d)ndarray
7	When this python code will execute, then which output will come? import numpy as np L1=[1,2] L2=[3,4] arr = np.array([L1, L2]) print(arr[1, 0]) **(a) 1 (b) 2 (c) 3 (d) 4**
8	from the below python code display the value of first element of array and display number 35 from an array import numpy as np Array=np.array([3,6,9,12,15,18,36,35,38]) print(Array[ ],Array[ ]) (a) [1],[7] (b) [1],[8] (c) [0],[8] (d)[0],[7]
9	Find the output of the below code: import numpy as np Array=np.array([3,6,3,3,15,18,36,35,38]) print (Array [:4])

	(a) [3 6 3 3] (b) [3,6,3,3,15] (c) [3,6,3,3] (d)[3,15,18,36,35,38]
10	Which of the following is correct way to import the Numpy module in your program? (a) import numpy (b) import numpy as np (c) from numpy import * (d) All of the above
11	What will be the output of following code? import numpy as np A=np.array([24,46,57,14,68,34,89,92]) print(A[7:3:-1]) print(A[2:6]) a) [92 89 34 68] [57 14 68 34] b) [92 34 89 68] [14 57 68 34] c) [92 68 34 89] [14 68 57 34] d) [92 34 68 89] [34 14 68 57]
12	Find out the Correct Statement: (a) Python List occupy less space than a NumPy array. (b) NumPy array can contain elements of non-homogenous type. (c) We cannot change the size of NumPy array after the store in memory (d) All of the above.

Answer (Multiple Choice Question and Fill in the Blanks)

1	(b) Numerical Python
2	(d) ndarray
3	True
4	(b) 2
5	import numpy as **np** Array=np.**array**([1,2,3,5,6]) print(Array)
6	(b)ndim
7	(c) 3 Explanation: arr value will be [[1 2] [3 4]] print(arr[1,0]) To access elements from 2-D arrays we can use comma separated integers representing the dimension and the index of the element.
8	(d) [0],[7]
9	(a) [3 6 3 3] Explanation: print (Array [:4]) This statement starts printing element from index position 0 and will print upto index position 4-1=3 as its increment by 1 (default increment).
10	(d) All of the above
11	(a) [92 89 34 68] [57 14 68 34] Explanation: print (A [7:3: -1]) This statement starts printing element from index

	position 7 and display upto index position (3-(-1))=4 as its decrement by -1
12	(c) We cannot change the size of NumPy array after the store in memory.

Assertion and Reasoning Questions:

Choose correct option for given Assertion (A) and Reasoning (R)

a) Both A and R are true, and R is the correct explanation of A.
b) Both A and R are true, but R is not the correct explanation of A.
c) A is true, but R is false.
d) A is false, but R is true.

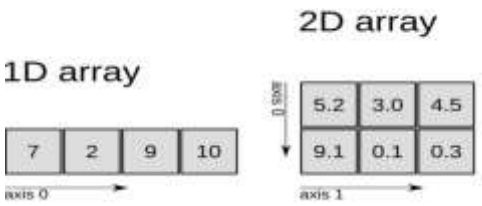
1	**Assertion(A):** NumPy is an open-source numerical python library. NumPy contain a multi-dimensional array and matrix data structure **Reason(R):** Array contain set of homogeneous values stored under one name. Array can be one dimensional, 2 D or multi-dimensional
2	**Assertion(A):** in python list values are separated by comma but in NumPy array value are separated by space at the time of display of value. It only used for distinguish purpose. **Reason(R):** both consume same size of memory
3	**Assertion(A):** for creating a list import of NumPy is required and for creating ndarray import NumPy library is not required **Reason(R):** ndarray cannot be created without importing NumPy but list can be creating without importing NumPy library

Answer(Assertion and Reasoning Questions)

1	(a) Both A and R are true, and R is the correct explanation of A.
2	(b) Both A and R are true, but R is not the correct explanation of A.
3	(d) A is false, but R is true.

Short Questions with Answer

1	What is the difference between Array and List
	Answer: **Difference between List and Array:** • Arrays only hold data of same data type, while list can hold the data of mixed data types (like int, float etc) • Using of Arrays to store data is more memory efficient and help to perform fast calculation and operations. • List is a part of core python, Array/ndarray is a part of NumPy Library • List take more space and more time of memory to store the elements because of different data types but Array takes less time and less space of memory to store elements because of same data type of elements of Array
2	How to create an Array of list in python using NumPy? Explain with suitable example.

	Answer: **NumPy Arrays** are grid-like structures similar to lists in Python. NumPy array can be created by converting a regular Python list into an array using the **np.array()** function. import numpy as np List1= [10,50,90,130] Arr1=np.array(List1) print (List1) # will display list [10,50,90,130] print (Arr1) # will display array of list [10 50 90 130]
3	What is an ndarray. What is use of Array in python code.
	Answer: **Ndarray** is one of the most important classes in the NumPy python library. It is basically a multidimensional or n-dimensional array of fixed size with homogeneous elements (i.e., the data type of all the elements in the array is the same) **Example** in this example 1-D array shape size is (1,4) and 2-D array shape size is (2,3) 
4	Write a program to make 1-D array and 2-D array list
	Answer: **1-D array of list** import numpy as np arr = np.array([1, 2,3, 4]) print(arr) **output:** [1 2 3 4] **2-D array of list** import numpy as np arr = np.array([[1, 2,3, 4], [5,6,7,8]]) print(arr) **output: [**[1 2 3 4] [5 6 7 8]**]**
5	What are some of the advantages of using NumPy arrays over Python lists?
	Answer: NumPy arrays offer several advantages, including: faster performance for numerical operations due to vectorization, more efficient memory usage, and a wide range of mathematical functions and operations. Array also helpful in Optimize the data that help CPU to perform operation accurately and effectively. NumPy is designed for data analysis and manipulation, offering a wide range of functions for sorting, searching, and statistical analysis.
6	Given a list L=[1,2,3,4] and an ndarray N having elements [1 2 3 4] . What will be the result produced by the following statements? (a) L*2 (b) N*3 (c) L+L (d) N+N
	Ans: **(a) [1,2,3,4,1,2,3,4]** **(b) [3 6 9 12]**

	(c) [1,2,3,4,1,2,3,4] **(d) [2 4 6 8]**

Case Based Question with Answer

1	store the following data: 2.5 19 0 3.4 18 7 10.5 19.2 5 (a) Use nested Python lists to create a 2-D array called array1 having 3 rows and 3 columns and (b) Write python code and find the size of the array1 (c) Write python code to find the datatype that store the elements in to the memory (d) Write python code to find out the shape of the nested list as per the list elements in array 1
	Answer: (a)

	Python code	**Output**
(a)	import numpy as np array1=np.array([[2.5,19,0],[3.4,18,7],[10.5,19.2,5]]) print(array1)	[[2.5 19. 0.] [3.4 18. 7.] [10.5 19.2 5.]]
(b)	print(array1.size)	9
(c)	print(array1.dtype)	float64
(d)	print(array1.shape)	(3, 3)

Web links:

1. Syllabus for Higher Secondary Stage prescribed by NCERT:

https://ncert.nic.in/pdf/syllabus/IPHSS.pdf

2. CBSE Syllabus Class XI – IP (065) Session 2025-2026

https://cbseacademic.nic.in/web_material/CurriculumMain26/SrSec/Informatics_Practices_SrSec_2025-26.pdf

3. NCERT Publication Book for IP (Class XI)

https://ncert.nic.in/textbook.php?keip1=0-8

4. Digital Text Book Class XI- IP on Diksha Portal

https://diksha.gov.in/resources/play/collection/do_313033532498550784178 5?contentType=TextBook