Chapter 1.

Q1. Overview of Python language ?

Ans. Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility.

Key Features:-

1. Easy to learn and use
2. High-level syntax
3. Interpreted language
4. Object-oriented programming (OOP)
5. Dynamic typing
6. Large standard library
7. Cross-platform compatibility
8. Extensive community support

History:-

1. Created by Guido van Rossum in 1991
2. First released in 1991
3. Version 3.x released in 2008

Applications:-

1. Web development (e.g., Django, Flask)
2. Data analysis and science (e.g., NumPy, Pandas)
3. Artificial intelligence and machine learning (e.g., TensorFlow, Keras)
4. Automation and scripting
5. Game development
6. Scientific computing
7. Education

Data Types:-

1. Integers
2. Floats
3. Strings
4. Lists
5. Tuples
6. Dictionaries
7. Sets
8. Boolean

Control Structures:-
1. If-else statements
2. For loops
3. While loops
4. Break and continue statements
5. Try-except blocks

Functions and Modules:-
1. User-defined functions
2. Built-in functions
3. Modules (e.g., math, statistics)
4. Packages (e.g., NumPy, Pandas)

Object-Oriented Programming (OOP):-
1. Classes and objects
2. Inheritance
3. Polymorphism
4. Encapsulation

Libraries and Frameworks:-
1. NumPy for numerical computing
2. Pandas for data manipulation
3. Matplotlib and Seaborn for data visualization
4. Scikit-learn for machine learning
5. Django and Flask for web development

Advantages:-
1. Easy to learn
2. Fast development
3. Large community
4. Cross-platform
5. Extensive libraries

Disadvantages:-
1. Slow performance
2. Limited support for parallel processing
3. Limited support for functional programming

Q2. Programming Constructs ?

Ans. Programming constructs are the building blocks of programming languages, used to write efficient, readable, and maintainable code. Here's an overview:

Sequence Constructs:-

1. Assignment Statements (e.g., x = 5)
2. Print Statements (e.g., print("Hello"))
3. Variable Declarations (e.g., int x;)

Control Flow Constructs:-

1. Conditional Statements:
   - If-Else Statements (e.g., if x > 5: print("Greater"))
   - Switch Statements (e.g., switch (x) {case 1: print("One")})
2. Loops:
   - For Loops (e.g., for i in range(5): print(i))
   - While Loops (e.g., while x < 5: print(x); x += 1)
   - Do-While Loops (e.g., do {print(x); x += 1} while (x < 5))
3. Jump Statements:
   - Break Statements (e.g., break;)
   - Continue Statements (e.g., continue;)
   - Return Statements (e.g., return x;)

Data Structures:-

1. Arrays (e.g., int[] scores = {1, 2, 3})
2. Lists (e.g., [1, 2, 3])
3. Tuples (e.g., (1, 2, 3))
4. Dictionaries (e.g., {"name": "John", "age": 30})
5. Sets (e.g., {1, 2, 3})

Functions and Modules:-

1. Function Definitions (e.g., def greet(name): print("Hello, " + name))
2. Function Calls (e.g., greet("John"))
3. Modules (e.g., import math; math.sqrt(4))
4. Lambda Functions (e.g., lambda x: x**2)

Object-Oriented Programming (OOP) Constructs:-

1. Classes (e.g., class Person {name; age;})
2. Objects (e.g., Person john = new Person("John", 30))

3. Inheritance (e.g., class Employee extends Person {salary;})
4. Polymorphism (e.g., method overriding)
5. Encapsulation (e.g., private variables)

Error Handling Constructs:-
1. Try-Catch Blocks (e.g., try {x = 5/0} catch (Exception e) {print(e)})
2. Throw Statements (e.g., throw new Exception("Error"))
3. Finally Blocks (e.g., finally {print("Cleanup")})

Other Constructs:-
1. Pointers (e.g., int* x = &y)
2. Generics (e.g., List<T> list = new List<T>())
3. Annotations (e.g., @Override)

Notes:- These programming constructs form the foundation of programming languages.

Q3. Data Structures like lists, dictionaries, tuples, sequences and their manipulations.

Ans. Here's an overview of data structures in Python, including lists, dictionaries, tuples, and sequences, along with their manipulations:

Lists:-
- Ordered collection of items
- Defined using square brackets []
- Elements can be of any data type
- Indexable and mutable

Example:- my_list = [1, 2, 3, 4, 5]

List Manipulations:-
- Indexing: my_list[0] returns 1
- Slicing: my_list[1:3] returns [2, 3]
- Append: my_list.append(6) adds 6 to the end
- Insert: my_list.insert(2, 7) inserts 7 at index 2
- Remove: my_list.remove(4) removes 4
- Sort: my_list.sort() sorts the list

Dictionaries:-
- Unordered collection of key-value pairs
- Defined using curly brackets {}
- Keys must be unique and immutable
- Values can be of any data type

Example:- my_dict = {"name": "John", "age": 30}

Dictionary Manipulations:-
- Accessing: my_dict["name"] returns "John"
- Updating: my_dict["age"] = 31 updates the value
- Adding: my_dict["city"] = "New York" adds a new key-value pair
- Removing: del my_dict["age"] removes the key-value pair
- Iterating: for key, value in my_dict.items(): iterates over key-value pairs

Tuples:-
- Ordered, immutable collection of items
- Defined using parentheses ()
- Elements can be of any data type

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

Tuple Manipulations:-
- Indexing: my_tuple[0] returns 1
- Slicing: my_tuple[1:3] returns (2, 3)
- No append, insert, or remove methods (immutable)

Sequences:-
- General term for lists, tuples, and strings
- Supports indexing, slicing, and concatenation

Example:-
my_string = "hello"
my_list = [1, 2, 3]
my_tuple = (4, 5, 6)

Sequence Manipulations:-
- Indexing: my_string[0] returns "h"
- Slicing: my_list[1:3] returns [2, 3]
- Concatenation: my_string + " world" returns "hello world"
- Repetition: my_list * 2 returns [1, 2, 3, 1, 2, 3]

Q4. Python Function ?

Ans. A function is a block of code which only runs when it is called.
You can pass data, known as parameters, into a function. A function can return data as a result.

Function Syntax:-

```
def function_name(parameters):
    # function body
```

Function Components:-

1. Function Name: Unique name for the function.
2. Parameters (Arguments): Values passed to the function.
3. Function Body: Code executed when the function is called.
4. Return Statement (Optional): Specifies the value returned by the function.

Function Types:-

1. Built-in Functions: Provided by Python (e.g., len(), print()).
2. User-defined Functions: Created by developers.
3. Lambda Functions: Anonymous functions (e.g., lambda x: x**2).
4. Generator Functions: Return iterators (e.g., yield).

Function Features:-

1. Function Arguments:
   - Positional arguments
   - Keyword arguments
   - Default argument values
   - Variable-length arguments (*args, **kwargs)
2. Function Return Values:
   - Single value
   - Multiple values (tuples, lists)
   - None (no return value)
3. Function Scope:
   - Local variables
   - Global variables

- Nonlocal variables

Function Examples:-

```
# Simple function
def greet(name):
    print(f"Hello, {name}!")

# Function with return value
def add(x, y):
    return x + y

# Lambda function
double = lambda x: x * 2

# Function with variable-length arguments
def sum_numbers(*numbers):
    return sum(numbers)
```

Chapter 6 (Function)

Q1. What is Function ?
Ans. A function is a block of code which only runs when it is called.
You can pass data, known as parameters, into a function. A function can return data as a result.

A function is a set of statements that take inputs, do some specific task and produces output.
Note: In other languages function are known as methods, procedores, subroutines etc.

Advantages of Python Function:-

- By using Function, we can avoid rewriting same code again and again in a program.
- We can call function on any place in any number of times in the program.
- We can track a large python program easily when it is divided into multiple function.

Types of Functions:-

1. **Built-in Functions:-**
2. **User define Functions:-**

1. Built-in function:-

The functions which are coming with Python software automatically are called **Built-in functions.** This is functions in Python programming language are called as **library function.** They are loaded automatically as the interpreter start and are always available.

For example:-

print(), and input() for I/O, number conversion functions int(), float(), complex(),data type conversions list(), tuple(), set(), etc.

2. User defined function:-

The function which are developed by programmers according to your requirement are called user defined functions.

Note:- while creating functions we can use two keywords like **def** (compulsory) and **return** (optional).

Creating a function:-

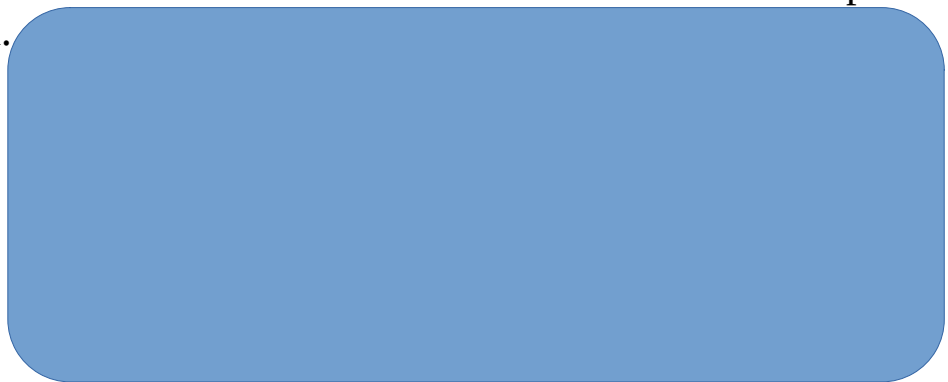
These are the basic steps in creating user-defined functions:

Step 1: To define your own Python function, you use the ‘def’ keyword before its name.

Step 2: Write the program arguments inside the opening and closing parentheses of the function, and end the declaration with a **colon.**

Step 3: Add the program statements to be executed. The statements inside the body of the function must be equally indented.

Step 4: End the function with/without return statement. A Python may optionally return a value. This value can be result that it produced on its execution.



```
def function_name(para):  => function header
    '''doc string'''      =>
    ...                   => function
    ...                   => body
    return statement(s)   =>
```

Calling a function:-

In python, a function a must be defined before the function calling otherwise the python interpreter gives an error. Once the function is defined, we can call it from another function or the python prompt. To call the function, use the function name before the parentheses.

Example:-

```
def wishing():
    print(" ..... ")
    print(" Welcome to Function ")
    print(" ..... ")
wishing()  # To call a Function
```

OUTPUT:- A simple function that prints the message is given below.

```
.....
Welcome to Function
.....
```

Example 2: A simple Python function to check whether x is even or odd.

```
def is_even():
   if x % 2 == 0:
      print("Is Even Number")
   else:
```

```
        print("Is Odd Number")
x=int(input("Enter a Number="))
is_even()
```

OUTPUT:-

Enter a Number = 4

Is Odd Number