**Basic Python Concepts:**

1. **Explain the difference between list, tuple, and set in Python. Provide examples of each**

**Answer:**

Difference between List, Tuple, and Set in Python

In Python, `list`, `tuple`, and `set` are data structures used to store collections of items, but they have distinct characteristics.

**1.List**

Definition: An ordered, mutable collection of elements. Allows duplicates.

Syntax: Defined using square brackets `[]`.

Mutability: Lists are mutable, meaning their elements can be modified.

Ordering: Lists maintain order, and elements are indexed starting from 0.

Example:

python

my\_list = [1, 2, 3, 'apple']

my\_list.append('orange') # Adds a new item

**2.Tuple**

Definition: An ordered, immutable collection of elements. Allows duplicates.

Syntax: Defined using parentheses `()`.

Mutability: Tuples are immutable, so their elements cannot be modified after creation.

Ordering: Tuples maintain order, like lists.

Example:

python

my\_tuple = (1, 2, 3, 'apple')

**3.Set**

Definition: An unordered, mutable collection of unique elements. Does not allow duplicates.

Syntax: Defined using curly braces `{}` or the `set()` function.

Mutability: Sets are mutable, but elements must be immutable.

Ordering: Sets do not maintain any order.

Example:

python

my\_set = {1, 2, 3, 'apple', 'banana', 'apple'} # Duplicate 'apple' is removed

**Key Differences:**

| **Feature**  | **List** | **Tuple** | **Set** |

|----------------|------------------|------------------|---------------|

| Syntax | `[ ]` | `( )` | `{ }` |

| Mutability | Mutable | Immutable | Mutable |

| Order | Ordered | Ordered |Unordered |

| Duplicates | Allowed | Allowed | Not allowed |

1. **How are functions defined and called in Python? Write a function to find the factorial of a number**.

**Answer:**

In Python, a function is defined using the `def` keyword, followed by the function name, parentheses `()`, and a colon `:`. The function body contains indented statements that are executed when the function is called.

**Function Definition Syntax: Recursion**

def function\_name(parameters):

# Function body

return result

- `def`: This keyword defines a function.

- `function\_name`: The name of the function.

- `parameters`: The inputs to the function (optional).

- `return`: This statement is used to return a result from the function (optional).

**Calling a Function:**

Once a function is defined, it can be called by using its name followed by parentheses. If the function takes arguments, they are passed inside the parentheses.

**Example:** Finding the Factorial of a Number

The factorial of a number `n` is the product of all positive integers less than or equal to `n`. For example, the factorial of `5` is `5! = 5 × 4 × 3 × 2 × 1 = 120`.

Here’s how to define and call a function that calculates the factorial of a number:

**Python Code:**

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n factorial(n - 1)

**#Calling the function**

result = factorial(5)

print("Factorial of 5 is:", result)

**Explanation:**

The function `factorial(n)` takes one argument `n`.

It uses recursion, where the function calls itself with `n-1`, until `n` is 0 or 1.

The base case returns `1` when `n` is `0` or `1` because `0! = 1! = 1`.

When `n = 5`, the function recursively multiplies `5 4 3 2 1`.

1. **What are comments in python?**

**Answer:**

Comments in Python are used to make code more understandable and are ignored by the interpreter.

**Types of Comments:**

1. Single-line Comments:

- Start with `#` and extend to the end of the line.

# This is a single-line comment

2. Multi-line Comments:

- You can use consecutive `#` or triple quotes (`'''` or `"""`) for multi-line comments.

'''

This is a multi-line comment

'''

**Purpose:** Comments are used to explain code, improve readability, and temporarily disable code during debugging.

1. **Explain Dictionary in python.**

**Answer:**

A dictionary in Python is an unordered, mutable collection that stores data in key-value pairs. Keys are unique and immutable, while values can be of any data type.

**Syntax:**

my\_dict = {

'name': 'Alice',

'age': 25,

'city': 'New York'

}

- Keys: `'name'`, `'age'`, `'city'` (must be unique)

- Values: `'Alice'`, `25`, `'New York'`

**Key Features:**

- Accessing Values: Use keys to access values.

print(my\_dict['name']) # Output: Alice

**Mutability:** You can add, modify, or remove key-value pairs.

my\_dict['age'] = 26 # Modify value

**Use Case:** Dictionaries are used for fast lookup and mapping relationships (like storing data by IDs).

1. **Explain Type Casting / Casting in python.**

**Answer:**

Type casting (or type conversion) in Python is the process of converting one data type into another. It can be done explicitly by using predefined functions.

**Common Type Casting Functions:**

1. int(: Converts a value to an integer.

x = int(3.5) # Output: 3

2. float(: Converts a value to a floating-point number.

y = float(5) # Output: 5.0

1. str(): Converts a value to a string.

z = str(10) # Output: '10'

4. list(), tuple(), set(): Converts to list, tuple, or set respectively.

my\_list = list((1, 2, 3)) # Output: [1, 2, 3]

**Use Case:** Type casting is useful when combining or processing different data types.

**6.Explain Operators In Python.**

Operators in Python are symbols used to perform operations on variables and values.

1. Arithmetic Operators: Perform mathematical operations.

- `+` (Addition), `-` (Subtraction), `` (Multiplication), `/` (Division), `%` (Modulus), `` (Exponentiation), `//` (Floor Division)

x = 10 + 5 # Output: 15

2. Comparison Operators: Compare two values and return `True` or `False`.

- `==`, `!=`, `>`, `<`, `>=`, `<=`

x = 5 == 5 # Output: True

3. Logical Operators: Combine conditional statements.

- `and`, `or`, `not`

x = (5 > 3) and (10 > 5) # Output: True

4. Assignment Operators: Assign values to variables.

- `=`, `+=`, `-=`, `=`, `/=`, etc.

x = 5

x += 3 # Output: 8

5. Bit-wise Operators: Perform bit-level operations.

- `&` (AND), `|` (OR), `^` (XOR), `~` (NOT), `<<` (Left Shift), `>>` (Right Shift)

6. Membership Operators: Check if a value is in a sequence.

- `in`, `not in`

x = 5 in [1, 2, 5] # Output: True

7. Identity Operators: Check if two objects are the same.

- `is`, `is not`

x = 5 is 5 # Output: True

**Use Case:** Operators are used for performing calculations, comparisons, logical operations, and more in Python programs.

**7.Explain Python `try-except`**

The `try-except` block in Python is used for handling exceptions (errors) that may occur during program execution, preventing crashes.

**Syntax:**

try:

# Code that may raise an exception

except ExceptionType:

# Code to handle the exception

**Example:**

try:

x = 10 / 0

# This will raise a ZeroDivisionError except ZeroDivisionError:

print("Cannot divide by zero!")

In this example, instead of crashing, the program prints an error message when a division by zero occurs.

**Multiple Exception Handle:**

try:

print(x)

except NameError:

print("Variable x is not defined")

except:

print("Something else went wrong")

**Use Case:** `try-except` is used to handle runtime errors gracefully, allowing the program to continue running or provide helpful error messages.

**Finally Block:**

try:

print(x)

except:

print("Something went wrong")

finally:

print("The 'try except' is finished")

**Important Question And Answer:**

1. **Four key features of Python, summarized:**

**Answer:**

1. Readable and Simple Syntax: Python’s clear and easy-to-read syntax makes it straightforward to write and understand code.

2. Dynamically Typed: Variables do not need explicit type declarations; their type is determined at runtime.

3. Interpreted Language: Python code is executed line-by-line, allowing for interactive testing and quick development.

1. Extensive Standard Library: Python includes a large collection of built-in modules and functions for various tasks, reducing the need for additional coding.
2. **State the floor division.**

**Answer:**

Floor Division in Python is used to divide two numbers and return the largest integer less than or equal to the result. It’s represented by the `//` operator.

**Use:**

- Returns an Integer: It performs division and discards the remainder, giving the integer part of the quotient.

- Useful for Indexing and Rounding: Ideal for scenarios where you need whole numbers, such as indexing lists or rounding down.

**Example:**

result = 7 // 3 # Output: 2

Here, `7 // 3` equals `2`, which is the largest integer less than or equal to the actual division result `2.3333...`.

**3.Write a python Program to print table of number or print no from 1 to 50 print even and odd no**

**Answer:**

1. **Print Multiplication of table of number:**

**Code:**

def print\_table(number):

for i in range(1, 11): # From 1 to 10

print(f"{number} x {i} = {number i}")

# Example usage

number = 5 # Change this to any number you want

print\_table(number)

1. **Print Numbers from 1 to 50, Separating Even and Odd**

**Code:**

def print\_even\_odd():

print("Even numbers:")

for i in range(1, 51):

if i % 2 == 0:

print(i, end=" ")

print("\nOdd numbers:")

for i in range(1, 51):

if i % 2 != 0:

print(i, end=" ")

# Call the function to execute

print\_even\_odd()

**4.Fibonacci series in reverse order**

**Code:**

def Fibonacci\_series(n):

# Generate the Fibonacci series

fib = [0, 1]

while len(fib) < n:

fib.append(fib[-1] + fib[-2])

# Print the Fibonacci series in reverse order

print("Fibonacci series in reverse order:")

for num in reversed(fib):

print(num, end=" ")

# Example usage

n = 10 # Change this value to get more or fewer numbers

Fibonacci\_series(n)

1. **Explain 4 Math and 4 String function:**

**Math Functions:**

1. `abs(x)`: Returns the absolute value of `x`, which is the non-negative value of `x`.

```python

abs(-5) # Output: 5

```

2. `pow(x, y)`: Returns `x` raised to the power of `y` (i.e., `x^y`).

```python

pow(2, 3) # Output: 8

```

3. `round(x, n)`: Rounds the number `x` to `n` decimal places. If `n` is not provided, it rounds to the nearest integer.

```python

round(3.14159, 2) # Output: 3.14

```

4. `sqrt(x)`: Returns the square root of `x`. Requires importing the `math` module.

```python

import math

math.sqrt(16) # Output: 4.0

```

**String Function**s

1. `len(s)`: Returns the length of the string `s`.

```python

len("hello") # Output: 5

```

2. `upper()`: Converts all characters in the string to uppercase.

```python

"hello".upper() # Output: "HELLO"

```

3. `lower()`: Converts all characters in the string to lowercase.

```python

"HELLO".lower() # Output: "hello"

```

4. `replace(old, new)`: Replaces occurrences of sub-string `old` with `new` in the string.

```python

"hello world".replace("world", "Python") # Output: "hello Python"

```