Python assignment(module 7)

1) What are the types of Applications?

Ans. Types of application in python.

Python can be used to create many types of applications. It is a versatile language.

- Desktop GUI applications:

  These are software applications with a graphical user interface.

  Ex: calculator, notepad, desktop games.

- Data science and analytics applications:
  Used to analyze and visualize data. Liabraries used in this application is pandas, numpy.
  Ex: Sales analysis.

- Machine learning and AI applications:
  Used to create smart or intelligent systems.
  Ex: face recognition, chatbots.

- Scientific and numeric application:
  Used to solve scientific or mathematical problems.
  Ex: equation solver

- Cybersecurity and ethical hacking tools:
  Used for security testing and ethical hacking purposes:

Ex: password cracker.

- Game development:
  Python can be used to create simple 2D games.
  Ex: snake game

- Networking applications:
  Used to handle computer networking tasks.
  Ex: chat apps, file transfer apps.

2) What is programing?

Ans. **Programming** means giving instructions to a computer to perform specific tasks. These instructions are written in a **programming language** — like **Python**.

Just like we speak English or Hindi, computers understand code. So, **programming in Python** means writing Python code to solve problems, build software, or control a computer.

3) What is Python?

Ans. **Python** is a powerful, high-level, interpreted programming language that is used for a wide range of applications such as **web development, data science, artificial intelligence, machine learning, automation, and more**.

It was created by **Guido van Rossum** and was first released in **1991**.

4) Write a Python program to check if a number is positive, negative or zero.

Ans. num = 76

```
if num > 0:
    print("positive number")
elif num == 0:
    print("zero")
else:
    print("negative number")
```

o/p = positive

5)Write a Python program to get the Factorial number of given numbers.

Ans.

```
num = int(input("enter the number: "))
fact = 1
if num < 0:
    print("fact of 0 does not exists")
elif num == 0:
    print("factorial of 0 is", 1)
else:
    for i in range(1, num + 1):
        fact = fact * i
```

```
    print("the factorial of the given number is", fact)
```

o/p = 5

the factorial is 120

6)Write a Python program to get the Fibonacci series of given range.

Ans.

```
n = int(input("enter the number of fibonacci series"))
a = 0
b = 1
if n <= 0:
    print("enter a positive integer")
elif n == 1:
    print("fibonacci series upto 1")
    print(a)
else:
    print("fibonacci series")
    for i in range(n):
        print(a, end=" ")
        a, b = b, a + b
```

o/p = enter the number of fibonacci series 10

fibonacci series

0,1,1,2,3,5,8,13,21,34

7)How memory is managed in Python?

Ans. In python the memory management happens automatically, meaning you don't have to manually allocate and free memory like in c or c++. Python has a memory management system that works in three main parts:

1. Private Heap Memory:

Python has a private heap where all objects and variables are stored.This heap memory is controlled by the Python interpreter can't directly access it.

2. Memory Manager:

The memory manager decides where in the heap to allocate space for a new object and when to free it.It takes care of memory allocation (reserving space) and deallocation (freeing space).

3. Garbage Collection (GC):

Python uses a garbage collector to automatically clean up unused objects from memory.

It mainly works with:

a) Reference Counting

Each object has a reference count (how many variables point to it).

When reference count becomes 0, the object is deleted.

b) Generational Garbage Collection:

Objects are divided into 3 generations based on their lifespan.

Long-living objects are checked less frequently to save performance. This makes garbage collection faster.

8)What is the purpose continuing statement in python?

Ans. The **continue statement** in Python is used **inside loops** (for or while) to **skip the rest of the code** inside the loop **for the current iteration** and move to the **next iteration** of the loop.

The purpose of the continue statement is to **skip** certain values or steps **without stopping** the entire loop.

9) Write python program that swap two number with temp variable and without temp variable.

Ans. # with temp variable

```
a = int(input("Enter first number (a"))

b = int(input("Enter second number b"))

temp = a

a = b

b = temp

print("after swapping using a temp variable")

print("a=",a)
```

```
print("b=", b)
# without using temp variable
a = int(input("Enter first number a"))
b = int(input("Enter second number b"))
a,b = b,a
print("after swapping without using a temp variable")
print("a=",a)
print("b=", b)
```

10) Write a Python program to find whether a given number is even or odd, print out an appropriate message to the user.

Ans. **# ask the user for input number and check number is even or odd**

```
number = int(input("enter a number:23"))
if 2 == 0:
    print("the number is even.2")
else:
    print("the number is odd.")
```

o/p = enter the number[77]

the number is odd

11) Write a Python program to test whether a passed letter is a vowel or not.

Ans. letter = (input("enter a letter"))

```
if letter in "aeiou":
    print("letter is vowel")
else:
    print("letter is not vowel")
```

o/p = enter a letter u

letter is vowel

12) Write a Python program to sum of three given integers. However, if two values are equal sum will be zero.

Ans. a = int(input("enter first value"))

```
b = int(input("enter second value"))
c = int(input("enter third value"))
if a == b or b == c or c == a:
    print("sum is ", 0)
else:
    print("sum is ", a+b+c)
```

o\p = enter first value 44

enter second value 32

enter third value 11

sum is 87

13) Write a Python program that will return true if the two given integer values are equal or their sum or difference is 5.

Ans.
```
def check(a,b):
    if a == b or abs(a-b) == 5 or (a + b) == 5:
        return True
    else:
        return False
num1 = int(input("enter first number"))
num2 = int(input("enter second number"))
result = check(num1, num2)
print("result",result)
```
o/p = enter first number 4

enter second number 1

result true

14)Write a python program to sum of the first n positive integers.

Ans.
```
num = int(input("enter the number"))
value = 0
for i in range(1, num + 1):
    value = value + i
```

```
print("sum of n numbers", value)
```

o/p = enter the number 4

sum of n number 10

15)Write a Python program to calculate the length of a string.

Ans.

```
# length of string
p1 = input("enter a string")
print(len(p1))
```

o/p = enter a string(hiee Nisha)

10

16) Write a Python program to count the number of characters (character frequency) in a string.

Ans.

```
# Input from user
string = input("Enter a string")

# Create an empty dictionary to store character counts
char_freq = {}

# Loop through each character in the string
for char in string:
    if char in char_freq:
```

```
        char_freq[char] += 1  # Increase count if already exists
    else:
        char_freq[char] = 1   # Set count to 1 if first time
# Output the result
print("Character Frequency:")
for char, count in char_freq.items():
    print(f"'{char}': {count}")
```

17)What are negative indexes and why are they used?

Ans. Negative indexes in programming are used to access elements in sequences(like lists, strings, and tuples) from the end. In python , negative indexing starts with -1 for the last element, -2 for the second-to-last, and so on.

- Why are they used?
  - Convenience:

    It's often more convenient to access elements from the end.
  - Readability:

    It makes the code shorter and easier to understand.
  - Efficiency:

    In some cases, negative indexing can be slightly more efficient, particularly when dealing with dynamic lists where the length is not know in advance.

18)Write a Python program to count occurrences of a substring in a string.

Ans. my_string = input("enter the main string")

my_string = input("enter the substring")

count = main_string.count(sub_string)

print(f"the substring '{sub_string}' occurs {count} times in the main string")

o/p = enter the main string

enter the substring

19) Write a Python program to count the occurrences of each word in a Page 17 of 28 given sentence.

Ans. sentence = input("Enter a sentence: ")

Split sentence into words

words = sentence.split()

Count each word

for word in set(words):

print(f"'{word}': {words.count(word)}")

20) Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.

Ans.# input strings

```
a = input("Enter first string: ")
b = input("Enter second string: ")

# swap first two characters
result = b[:2] + a [2:] + " " + a[:2] + b[2:]
print(result)
```

21) Write a Python program to add 'in' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead if the string length of the given string is less than 3, leave it unchanged.

Ans. word = input("Enter a word: ")

```
if len(word) < 3:
    print(word)
elif word.endswith("ing"):
    print(word + "ing")
else:
    print(word + "ly")
```

o/p = enter the string = like

likely

22) Write a Python function to reverses a string if its length is a multiple of 4.

Ans. word = input("Enter a word: ")

```
if len(word) % 4 == 0:
    print("Reversed:", word[::-1])
else:
    print("Same:", word)
```

o/p = enter a word = cool

looc

23) Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string.

Ans. word = input("Enter a string")

```
if len(word) < 2:
    print("Result: '")  # empty string
else:
    result = word[:2] + word[-2:]
```

```
    print("Result:", result)
```

o/p = enter a string = Nisha

niha

24)Write a Python function to insert a string in the middle of a string.

Ans.
```
def insert_middle(main_str, insert_str):
    mid = len(main_str) // 2
    return main_str[:mid] + insert_str + main_str[mid:]


str1 = input("Enter the main string: ")
str2 = input("Enter the string to insert: ")


result = insert_middle(str1, str2)
print("Result:", result)py
```

o/p = enter main string= helloworld

enter substring= py

hellopyworld

25)What is List? How will you reverse a list?

Ans. A **List** in Python is a type of **data structure** that is used to **store multiple values** in a single variable. It is one of the most commonly used collection types in Python.

- How will you reverse a list:
  Reversing a list means **changing the order** of the elements so that the **last becomes first and first becomes last**.

- **Ways to Reverse a List:**
  - Built in method
  - Slicing method
  - Using a loop

26)How will you remove last object from a list?

Ans. Removing the last element from a list means deleting the items at the end of the list. Thin is often done to update or shorten the list , discard unwanted data or manage memory in certain programs.

There are many ways to remove last element from list in python.

- Using pop() method:
  Pop method removes and returns the last element of a list. It allows the users to directly modifies the original list and is useful when you need to both update the list and use the removed items.
- Using del operator:
  Del operator allows removal of list element by index. Which directly modifies the original list by removing its last item without returning it.
- **remove() method:**

- **Use:** Removes the **first occurrence of a specific value**.

- You need to **mention the value**, not the index.
- If the value is not found, it gives an error.

➢ **clear() method:**

- **Use:** Removes **all elements** from the list.
- List becomes **empty**.

27) Suppose list1 is [2, 33, 222, 14, and 25], what is list1 [- 1]?

Ans. list1[-1] means:
Access the last element of the list using a **negative index**.

- list1[0] → first element
- list1[1] → second element
- list1[-1] → **last element**

---

list1 = [2, 33, 222, 14, 25]

list1[-1] = **25**

28)Differentiate between append () and extend () methods?

Ans.

Feature / Point	**append()**	**extend()**
Purpose	Adds a **single element** to the end of the list	Adds **multiple elements** to the list

Feature / Point	append()	extend()
Input Type	Takes a **single item** (number, string, list)	Takes an **iterable** (like list, tuple, string)
How It Adds	Adds the item **as it is**	Adds each item from the iterable **one by one**
Change in List Length	List size increases by **1**	List size increases by **length of iterable**
Works with	Any data type (even a list as a single element)	Only iterable types (list, tuple, etc.)
Example Output	[1, 2].append([3, 4]) → [1, 2, [3, 4]]	[1, 2].extend([3, 4]) → [1, 2, 3, 4]
Use Case	Add **one item or object**	Add **multiple values at once**

29) Write a Python function to get the largest number, smallest num and sum of all from a list.

Ans. def analyze_list(numbers):

```
    largest = max(numbers)
    smallest = min(numbers)
    total_sum = sum(numbers)
```

```
    print("Largest number:", largest)
    print("Smallest number:", smallest)
    print("Sum of all numbers:", total_sum)


my_list = [22,44,22,1,3,4,88,96,23]
analyze_list(my_list)
```

o/p = largest number 96

smallest number 1

sum of total numbers 303

30)How will you compare two lists?

- Ans. **. Using == Operator**
  - Checks if **both lists are exactly the same**
  - Compares **elements and their order**
  - Returns True or False
- **Using != Operator**
  - Checks if two lists are **not equal**
  - Useful when you want to know if there's **any difference**
- **Using set() Function**
  - Converts both lists into sets
  - Compares only the **unique elements**, **ignores order**

- Doesn’t work well if there are **duplicate values**

➢ **Using sorted() Function**

- Sorts both lists and then compares
- Ignores order but checks if elements and frequency are the same
- Useful when you care about values, not position

➢ **Using Loops (Manual Comparison)**

- Compares **element-by-element**
- Good when you want to find **exact differences**
- Flexible for custom rules (like ignoring case, partial match, etc.)

➢ **Using collections.Counter()**

- Compares elements and their **frequency/count**
- Ignores order
- Works well for **lists with repeated elements.**

31) Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings.

Ans. words = ["abc", "xyz", "aba", "1221", "aa", "b"]

```
count = 0
for word in words:
```

```
    if len(word) >= 2 and word[0] == word[-1]:
        count += 1

print("Number of strings:", count)
```

o/p = number of strings 3

32) Write a Python program to remove duplicates from a list.

Ans.

```
my_list = [1, 2, 2, 3, 4, 4, 5]
unique_list = []

for item in my_list:
    if item not in unique_list:
        unique_list.append(item)

print("List without duplicates:", unique_list)
```

o/p = 1,2,3,4,5

33)Write a Python program to check a list is empty or not.

Ans.

```
list = []
if not l:
    print("list is empty")
else:
```

```
    print("list is not empty")
```

o/p = list is empty

34) Write a Python function that takes two lists and returns true if they have at least one common member.

Ans.
```
def common_member(list1, list2):
    for item in list1:
        if item in list2:
            return True
    return False


print(common_member([1, 2, 3], [4, 5, 6]))
print(common_member([1, 2, 3], [3, 4, 5]))
```

o/p = false

true

have common member true

35) Write a Python program to generate and print a list of first and last 5 elements where the values are square of numbers between 1 and 30.

Ans.
```
squares = [x**2 for x in range(1, 31)]


first_5 = squares[:5]
```

```
last_5 = squares[-5:]


print("First 5 squares:", first_5)
print("Last 5 squares:", last_5)
```

o/p = First 5 squares: [1, 4, 9, 16, 25]

Last 5 squares: [676, 729, 784, 841, 900]

36) Write a Python function that takes a list and returns a new list with unique elements of the first list.

Ans.
```
def unique_elements(my_list):
    unique = []
    for i in my_list:
        if i not in unique:
            unique.append(i)
    return unique


numbers = [1, 2, 2, 3, 4, 4, 5]
print("Unique items:", unique_elements(numbers))
```

o/p = Unique items: [1, 2, 3, 4, 5]

37)Write a Python program to convert a list of characters into a string.

Ans. char_list = ['h', 'e', 'l', 'l', 'o']

result = ''.join(char_list)

print("String:", result)

o/p = String: hello

38)Write a Python program to select an item randomly from a list.

Ans. import random

items = [10, 20, 30, 40, 50]

random_item = random.choice(items)

print("Random item:", random_item)

o/p = Random item: 40

39) Write a Python program to find the second smallest number in a list.

Ans. numbers = [5, 2, 8, 1, 9, 3]

```
numbers.sort()
second_smallest = numbers[1]

print("Second smallest number:", second_smallest)
```

o/p = Second smallest number: 2

40) Write a Python program to get unique values from a list.

Ans.
```
numbers = [1, 2, 2, 3, 4, 4, 5]
unique_values = list(set(numbers))
print(unique_values)
```

o/p = 1,2,3,4,5

41) Write a Python program to check whether a list contains a sub list.

Ans.
```
main_list = [1, 2, 3, 4, 5]
sub_list = [3, 4]

if str(sub_list)[1:-1] in str(main_list)[1:-1]:
    print("Sublist exists")
else:
    print("Sublist does not exist")
```

o/p = Sublist exists

42) Write a Python program to split a list into different variables.

Ans. **# Sample list**

my_list = [10, 20, 30]

 # Split into variables

a, b, c = my_list

print("a =", a)

print("b =", b)

print("c =", c)

43) What is tuple? Difference between list and tuple.

Ans. A **tuple** is a type of data structure in Python that:

- Is **ordered** (items are stored in a specific sequence).
- Is **immutable**, meaning you **cannot change, add, or remove** elements after it is created.
- Uses **round brackets** () to store data.
- Can store **different types of values**, like numbers, strings, etc.

Tuples are useful when you want to store a group of values that should **not be changed**.

Feature	List	Tuple
Brackets Used	Square brackets []	Round brackets ()
Mutability	Mutable (you can change it)	Immutable (you cannot change it)
Speed	Slower	Faster
Use Case	Use when values might change	Use when values must stay fixed
Memory	Takes more memory	Takes less memory
Functions/Methods	Many (append, pop, remove, etc.)	Few (only count and index)

44) Write a Python program to create a tuple with different data types.

Ans. my_tuple = (25, 4.5, "Python", True, None, [1, 2, 3], {'name': 'Alex'}, (9, 8))

```
print("This is a tuple with different data types:")
print(my_tuple)

print("\nEach element and its data type:")
for item in my_tuple:
```

```
    print(item, "→", type(item))
```

o/p = This is a tuple with different data types:

(25, 4.5, 'Python', True, None, [1, 2, 3], {'name': 'Alex'}, (9, 8))

Each element and its data type:

25 → <class 'int'>

4.5 → <class 'float'>

Python → <class 'str'>

True → <class 'bool'>

None → <class 'NoneType'>

[1, 2, 3] → <class 'list'>

{'name': 'Alex'} → <class 'dict'>

(9, 8) → <class 'tuple'>

45) Write a Python program to unzip a list of tuples into individual lists.

Ans. list_of_tuples = [(1, 'a'), (2, 'b'), (3, 'c')]

```
list1, list2 = zip(*list_of_tuples)

list1 = list(list1)
list2 = list(list2)
```

print(list1)

print(list2)

o/p = [1, 2, 3]

['a', 'b', 'c']

46) Write a Python program to convert a list of tuples into a dictionary.

Ans. **# List of tuples**

list_of_tuples = [('a', 1), ('b', 2), ('c', 3)]

Convert to dictionary using dict()

my_dict = dict(list_of_tuples)

Print the dictionary

print("Converted Dictionary:", my_dict)

47) How will you create a dictionary using tuples in python?

Ans. A **tuple** is an ordered, immutable collection.
If you have multiple tuples — where each tuple has exactly **two elements** (key, value) — you can convert them into a **dictionary**.

Ways to Create Dictionary from Tuples:

1. Using a List of Tuples:

Example:

```
data = [("name", "Alice"), ("age", 25), ("city", "Delhi")]
my_dict = dict(data)
```

2. Using a Tuple of Tuples:

Example:

```
data = (("a", 1), ("b", 2), ("c", 3))
my_dict = dict(data)
```

3. Using a Single Tuple Containing Two Tuples (Not common):

Example:

```
data = (("x", 10), ("y", 20))
my_dict = dict(data)
```

48) Write a Python script to sort (ascending and descending) a dictionary by value.

Ans. my_dict = {'a': 3, 'b': 1, 'c': 2}

```
# Sort by value (ascending)
asc = dict(sorted(my_dict.items(), key=lambda x: x[1]))

# Sort by value (descending)
desc = dict(sorted(my_dict.items(), key=lambda x: x[1], reverse=True))
```

```
print("Ascending:", asc)
print("Descending:", desc)
```

o/p = Ascending: {'b': 1, 'c': 2, 'a': 3}

Descending: {'a': 3, 'c': 2, 'b': 1}

49) Write a Python script to concatenate following dictionaries to create a new one.

Ans.

```
# Dictionaries to concatenate
dict1 = {1: "a", 2: "b"}
dict2 = {3: "c", 4: "d"}
dict3 = {5: "e", 6: "f"}

# Method 1: Using update()
result = {}
for d in (dict1, dict2, dict3):
    result.update(d)

print("Concatenated Dictionary:", result)
```

50) Write a Python script to check if a given key already exists in a dictionary.

Ans. **# Sample dictionary**

my_dict = {'name': 'Nisha', 'age': 20, 'city': 'Delhi'}

key_to_check = 'age'

 # Check if key exists

if key_to_check in my_dict:

    print(f"Yes, the key '{key_to_check}' exists in the dictionary.")

else:

    print(f"No, the key '{key_to_check}' does not exist in the dictionary.")

51) How Do You Traverse Through a Dictionary Object in Python?

Ans. Traversing (**looping**) through a **dictionary object** in Python means accessing each **key**, **value**, or both **key-value pairs** one by one. Here's how you can do it — in simple theory (without code):

- **Ways to Traverse a Dictionary:**

1. Using for loop to access only keys

- You can loop through the dictionary directly.

- It will return each key one by one.

2. Using .keys()

- This gives you all the keys of the dictionary.
- Useful when you only want keys.

3. Using .values()

- Returns only the values in the dictionary.
- Good when you only care about values.

4. Using .items()

- Returns key-value pairs (in tuple form).
- Best method to access both keys and values together.

5. Using for loop with enumerate() (optional)

- Helpful when you also want the index along with key-value.

52) How Do You Check the Presence of a Key in A Dictionary?

Ans. To check the **presence of a key** in a dictionary in Python, you can use the following simple **theoretical methods** (no code):

- **Ways to Check if a Key Exists in a Dictionary:**

1. Using in operator

- Most common and direct way.
- It checks if a particular key is present in the dictionary.

2. Using get() method

- This method returns the value for a key if it exists, otherwise it returns None.
- You can use it with a condition to check if the result is not None.

3. Using keys() method with in

- You can explicitly check if the key is in the list of dictionary keys.
- Useful when you want to make your intention more clear.

53) Write a Python script to print a dictionary where the keys are numbers between 1 and 15.

Ans. # Create dictionary with keys 1 to 15 and values as their squares

```
my_dict = {num: num**2 for num in range(1, 16)}


# Print the dictionary
print(my_dict)
```

54)Write a Python program to check multiple keys exists in a dictionary.

Ans. # Sample dictionary

```
my_dict = {'name': 'Nisha', 'age': 21, 'city': 'Delhi', 'course': 'Python'}

keys_to_check = ['name', 'city']

# Check if all keys exist
if all(key in my_dict for key in keys_to_check):
    print("All keys exist in the dictionary.")
else:
    print("Some keys are missing.")
```

o/p = All keys exist in the dictionary.

55) Write a Python script to merge two Python dictionaries.

Ans.
```
dict1 = {'a': 1, 'b': 2}
dict2 = {'c': 3, 'd': 4}

merged_dict = {}

for key, value in dict1.items():
    merged_dict[key] = value

for key, value in dict2.items():
```

```
    merged_dict[key] = value

print("Merged Dictionary:", merged_dict)
```

o/p = Merged Dictionary: {'a': 1, 'b': 2, 'c': 3, 'd': 4}

56) Write a Python program to map two lists into a dictionary Sample output: Counter ({'a': 400, 'b': 400,'d': 400, 'c': 300}).

Ans.

```
from collections import Counter

keys = ['a', 'b', 'c', 'd']
values = [400, 400, 300, 400]

result = Counter(dict(zip(keys, values)))

print(result)
```

o/p = Counter({'a': 400, 'b': 400, 'd': 400, 'c': 300})

57) Write a Python program to find the highest 3 values in a dictionary.

Ans.

```
my_dict = {'a': 100, 'b': 300, 'c': 250, 'd': 150, 'e': 400}

values = list(my_dict.values())
```

```
values.sort(reverse=True)

top_3 = values[:3]

print("Top 3 highest values:", top_3)
```

o/p = Top 3 highest values: [400, 300, 250]

58) Write a Python program to combine values in python list of dictionaries. Sample data: [{'item': 'item1', 'amount': 400}, {'item': 'item2', 'amount': 300}, o {'item': 'item1', 'amount': 750}] Expected Output: • Counter ({'item1': 1150, 'item2': 300})

Ans.

```
data = [
    {'item': 'item1', 'amount': 400},
    {'item': 'item2', 'amount': 300},
    {'item': 'item1', 'amount': 750}
]

result = {}

for entry in data:
    item = entry['item']
    amount = entry['amount']
```

```
    if item in result:
        result[item] += amount
    else:
        result[item] = amount

print(result)
```

o/p = {'item1': 1150, 'item2': 300}

59) Write a Python program to create a dictionary from a string. Note: Track the count of the letters from the string.

Ans. string = "pythonprogramming"

```
letter_count = {}

for char in string:
    if char in letter_count:
        letter_count[char] += 1
    else:
        letter_count[char] = 1

print(letter_count)
```

o/p = {'p': 2, 'y': 1, 't': 1, 'h': 1, 'o': 2, 'n': 2, 'r': 2, 'g': 2, 'a': 1, 'm': 2, 'i': 1}

60) Sample string: 'w3resource' Expected output: • {'3': 1,’s’: 1, 'r': 2, 'u': 1, 'w': 1, 'c': 1, 'e': 2, 'o': 1}

Ans. sample = 'w3resource'

char_count = {}

for char in sample:

    if char in char_count:

        char_count[char] += 1

    else:

        char_count[char] = 1

print(char_count)

o/p = {'w': 1, '3': 1, 'r': 2, 'e': 2, 's': 1, 'o': 1, 'u': 1, 'c': 1}

Selection deleted

61) Write a Python function to calculate the factorial of a number (a nonnegative integer).

Ans. def factorial(n):

    if n < 0:

        return "Factorial not defined for negative numbers"

```
    elif n == 0 or n == 1:
        return 1
    else:
        fact = 1
        for i in range(2, n + 1):
            fact *= i
        return fact

# Example
num = 5
print(f"Factorial of {num} is {factorial(num)}")
```

o/p = Factorial of 5 is 120

62) Write a Python function to check whether a number is in a given range.

Ans.

```
def check_in_range(num, start, end):
    if num in range(start, end + 1):
        return f"{num} is in the range ({start} - {end})"
    else:
        return f"{num} is NOT in the range ({start} - {end})"

print(check_in_range(5, 1, 10))
```

print(check_in_range(15, 1, 10))

o/p = 5 is in the range (1 - 10)

15 is NOT in the range (1 - 10)

63) Write a Python function to check whether a number is perfect or not.

Ans.
```
def check_perfect(num):
    total = 0
    for i in range(1, num):  # loop from 1 to num-1
        if num % i == 0:     # if i divides num perfectly
            total += i       # add i to total

    if total == num:
        print(num, "is a Perfect Number.")
    else:
        print(num, "is NOT a Perfect Number.")

# Example
check_perfect(6)
check_perfect(15)
```

o/p = 6 is a Perfect Number.

15 is NOT a Perfect Number.

64) Write a Python function that checks whether a passed string is palindrome or not

Ans.
```
def check_palindrome(text):
    if text == text[::-1]:  # reverse the string and compare
        print(text, "is a Palindrome.")
    else:
        print(text, "is NOT a Palindrome.")


# Example
check_palindrome("madam")
check_palindrome("hello")
```

o/p = madam is a Palindrome.

hello is NOT a Palindrome.

65) How Many Basic Types of Functions Are Available in Python?

Ans. 🔷 **There are 4 basic types of functions in Python:**

1. **Function with no arguments and no return value**
    - It doesn’t take any input from the user.
    - It doesn’t give back (return) any result.

- It just performs some action (like printing something).

---

2. **Function with arguments but no return value**
    - It takes input values (called arguments).
    - But it doesn’t return any result.
    - It might just display the result or perform a task.

---

3. **Function with no arguments but with return value**
    - It doesn’t take any input.
    - But it gives back (returns) a result.

---

4. **Function with arguments and return value**
    - It takes input values (arguments).
    - And it also returns a result.

66) How can you pick a random item from a list or tuple?

Ans. select **one random item** from a **list** or **tuple**.
Example:

- From ['apple', 'banana', 'mango'], pick any one fruit randomly.
- From ('red', 'green', 'blue'), pick any one color randomly.

Python has a built-in module called **random**, which helps in doing anything related to randomness, such as:

- Generating random numbers
- Picking random items from a collection (like list or tuple)

To pick a random item, Python provides a function called **choice()** inside the random module.

1. **use the random module**.
2. use the **choice() function**, and pass the list or tuple to it.
3. It **randomly selects** and gives you back **one item** from it.

67) How can you pick a random item from a range?

Ans. import random

```
# Define the range
r = range(10, 20)

# Pick a random item from the range
random_item = random.choice(r)

print(random_item)
```

68) How can you get a random number in python?

Ans. To get a **random number** in Python, use the **random module**, which provides several useful functions for generating random numbers.

- **Ways to Get a Random Number in Python:**

1. Random Integer (Whole Number)

- You can get a random **whole number** between two numbers (like 1 to 10).
- Function used: random.randint(start, end)
- Includes both start and end numbers.

---

2. Random Floating-point Number

- To get a **decimal number** between 0 and 1.
- Function used: random.random()
- It gives a random float like: 0.3456, 0.8712, etc.

---

3. Random Float in a Range

- To get a **decimal number between two values** (like 1.5 to 5.5).
- Function used: random.uniform(start, end).

69)How will you set the starting value in generating random numbers?

Ans. **random.seed(value)**

🔹 What is seed()?

- seed() sets the **starting point** for generating random numbers.
- This ensures that the **same random numbers** are generated **every time** you run the code.
- It is useful for **testing**, **debugging**, or **reproducibility**.

Why set a seed?

Without a seed:

- Random numbers change every time.

With a seed:

- Random numbers will be the **same each time you run** the program.

🔹 How it works (without code):

1. You set a seed using random.seed(5) (you can use any number).
2. After setting the seed, when you generate random numbers (e.g., using randint() or random()), they will always follow the **same pattern**.
3. This is useful to get **repeatable results**.

70) How will you randomize the items of a list in place?

Ans. **random.shuffle()**

🔷 What does "randomize in place" mean?

- It means **shuffling** or **mixing** the elements of a list in a **random order**.
- “In place” means the **original list itself is changed**, and no new list is created.

✅ Which function is used?

- Python provides a function called **shuffle()**.
- It is available inside the **random** module.
- So, you first use the random module, then use the shuffle() function.

🔷 What does shuffle() do?

- It **randomly rearranges** the items in the list.
- The original list gets **modified directly**.
- It does **not return anything**.
- Every time you use shuffle(), the list order will be **different**.

- If list is:
  `[1, 2, 3, 4, 5]`
- After using `random.shuffle()`, it might become:
  `[3, 5, 1, 4, 2]` (order is random)

71) What is File function in python? What are keywords to create and write file.

Ans. File functions in Python are used to **create**, **open**, **read**, **write**, and **close** files.

- These functions help in **storing data permanently** on the disk.
- The most commonly used function is **open()**, which is used to open or create a file.
- After opening a file, you can perform actions like **reading** or **writing** data.
- It's important to use **close()** after file operations to free up system resources.
- When using open(), you must specify the **mode** (a keyword string) that tells Python what to do with the file:

Mode	Meaning
`'w'`	Write mode (create or overwrite)
`'a'`	Append mode (add to end)
`'x'`	Create mode (error if exists)
`'r'`	Read mode (default)

72) Write a Python program to read an entire text file.

Ans.
```
# Open the file in read mode
with open('filename.txt', 'r') as file:
    # Read the entire content of the file
    content = file.read()
```

```
# Print the content
print(content)
```

73) Write a Python program to append text to a file and display the text.

Ans.
```
# Step 1: Append text to the file
with open("sample.txt", "a") as file:  # 'a' mode = append
    file.write("\nThis is a new line added to the file.")

# Step 2: Read and display the updated file content
with open("sample.txt", "r") as file:  # 'r' mode = read
    content = file.read()
    print("Updated File Content:")
    print(content)
```
o/p = Updated File Content:

This is a new line added to the file.

74) Write a Python program to read first n lines of a file.

Ans.
```
# Step 1: Ask user for number of lines to read
n = int(input("Enter number of lines to read: "))
```

```
# Step 2: Open the file and read first n lines
with open("sample.txt", "r") as file:
    for i in range(n):
        line = file.readline()
        if not line:  # Stop if file ends before n lines
            break
        print(line.strip())  # Remove extra newline characters
```

o/p = Enter number of lines to read: 3

This is a new line added to the file.

75) Write a Python program to read last n lines of a file.

Ans.

```
# Step 1: Ask user for number of lines to read from the end
n = int(input("Enter number of lines to read from end: "))

# Step 2: Open the file and read last n lines
with open("sample.txt", "r") as file:
    lines = file.readlines()  # Read all lines into a list
    last_lines = lines[-n:]   # Slice last n lines

# Step 3: Print the last n lines
```

```
for line in last_lines:
    print(line.strip())  # Remove extra spaces/newlines
```

o/p = Enter number of lines to read from end: 2

This is a new line added to the file.

76)Write a Python program to read a file line by line and store it into a list.

Ans. lines_list = []

```
with open("sample.txt", "r") as file:
    for line in file:
        lines_list.append(line.strip())

print(lines_list)
```

o/p = ['', 'This is a new line added to the file.']

77)Write a Python program to read a file line by line store it into a variable.

Ans. # Step 1:

```
content = ""
```

```
# Step 2:
file = open("sample.txt", "r")

# Step 3:
for line in file:
    content += line

# Step 4:
file.close()

# Step 5:
print("File Content:")
print(content)
```

o/p = File Content:

This is a new line added to the file.

78)Write a python program to find the longest words.

Ans.

```
sentence = input("Enter a sentence: ")
words = sentence.split()

longest_word = ""
```

```
for word in words:
    if len(word) > len(longest_word):
        longest_word = word

print("Longest word is:", longest_word)
```

79)Write a Python program to count the number of lines in a text file.

Ans.
```
filename = "yourfile.txt"  # Replace with your file name

with open(filename, 'r') as file:
    lines = file.readlines()
    count = len(lines)

print("Number of lines in the file:", count)
```

80)Write a Python program to count the frequency of words in a file.

Ans.
```
def count_words(filename):
    counts = {}
    with open(filename, 'r') as file:
```

```
        for line in file:
            words = line.lower().split()  # split line into words
            for word in words:
                word = word.strip('.,!?"\'')  # remove punctuation
                if word in counts:
                    counts[word] += 1
                else:
                    counts[word] = 1
    return counts

# Example:
file_name = 'sample.txt'  # change to your file
result = count_words(file_name)
print(result)
```

81)Write a Python program to write a list to a file.

Ans.

```
def write_list_to_file(filename, my_list):
    with open(filename, 'w') as file:
        for item in my_list:
            file.write(str(item) + '\n')  # write each item on a new
line
```

```
# Example usage:
my_list = ['apple', 'banana', 'cherry', 123, 45.6]
filename = 'output.txt'
write_list_to_file(filename, my_list)
print("List written to file successfully.")
```

82)Write a Python program to copy the contents of a file to another file.

Ans.
```
# Open the source file in read mode
source = open('source.txt', 'r')

# Open the destination file in write mode
destination = open('destination.txt', 'w')

# Read content from source and write it to destination
destination.write(source.read())

# Close both files
source.close()
destination.close()

print("File copied!")
```

83)Explain Exception handling? What is an Error in Python?

Ans. What is an Error in Python

An Error in Python is a problem that occurs during the execution of a program which causes it to stop running (crash) or behave unexpectedly. Errors can be:

- Syntax Errors: Mistakes in the code structure (e.g., missing colon, wrong indentation).
- Runtime Errors: Errors that happen while the program is running, like dividing by zero or accessing a non-existent file.
- Logical Errors: When the code runs but produces wrong results (not technically caught by Python as errors).

How Exception Handling Works in Python

You use the try-except block to catch exceptions:

```
try:
    # Code that might cause an exception
    x = 10 / 0
except ZeroDivisionError:
    # Code to run if ZeroDivisionError occurs
    print("You cannot divide by zero!")
```

- The code inside the try block is executed.
- If an exception occurs, Python looks for a matching except block.

- If found, the except block runs instead of crashing the program.

You can also handle multiple exceptions, use a generic except Exception for all exceptions, and add else and finally blocks for extra control.

84)How many except statements can a try-except block have? Name Some built-in exception classes:

Ans. You can have many except blocks after one try.

- Each except handles a different error.
- Python checks each except one by one to see which fits the error.

Example:

```
try:
    x = int("hello")  # This causes an error
except ZeroDivisionError:
    print("Can't divide by zero!")
except ValueError:
    print("Oops! Wrong value.")
except:
    print("Some other error happened.")
```

Some common built-in error names:

- ZeroDivisionError — when you divide by zero.
- ValueError — when the value is wrong (like converting "hello" to number).
- TypeError — when you use the wrong type (like adding number and text).
- IndexError — when you ask for a list item that doesn't exist.
- KeyError — when a dictionary key is missing.
- FileNotFoundError — when a file can't be found.
- AttributeError — when you use something that doesn't exist on an object.

85)When will the else part of try-except-else be executed?

Ans. The else part runs only when there is no error in the try part.

If everything in try is okay, then else runs.

But if try causes an error, else does NOT run.

86)Can one block of except statements handle multiple exception?

Ans. One except block can handle multiple exceptions in Python. You just need to group the exceptions inside a tuple.

Here's the syntax:

try:

```
    # some code that may raise exceptions
except (ExceptionType1, ExceptionType2):
    # handle both ExceptionType1 and ExceptionType2 here
```

Example:

```
try:
    x = int(input("Enter a number: "))
    result = 10 / x
except (ValueError, ZeroDivisionError) as e:
    print("Error occurred:", e)
```

In this example, if the user enters a non-integer (causing ValueError) or zero (causing ZeroDivisionError), the same except block will handle both.

87)When is the finally block executed?

Ans. The finally block in Python is executed always, no matter what — whether an exception was raised or not, whether it was caught or not.

Here’s when the finally block runs:

After the try block finishes (whether normally or due to an exception).

After any matching except blocks run (if an exception was caught).

Even if there's a return, break, or continue in the try or except blocks.

Even if an exception is not caught and propagates up.

88)What happens when „1"== 1 is executed?

Ans. When you execute the expression:

'1' == 1

it evaluates to False.

Why?

- '1' is a string (text type).
- 1 is an integer (numeric type).

In Python, the equality operator == compares both the value and the type in a way that they have to be compatible. A string '1' and an integer 1 are different types, so they are not considered equal.

89)How Do You Handle Exceptions with Try/Except/Finally in Python? Explain with coding snippets.

Ans. 1. Basic Try-Except

You put the risky code inside the try block. If an error occurs, Python jumps to the matching except block.

```
try:
    x = 10 / 0  # This will raise ZeroDivisionError
except ZeroDivisionError:
    print("You can't divide by zero!")
```

2. Multiple Except Blocks

You can handle different exceptions separately.

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
except ValueError:
    print("That's not a valid number!")
except ZeroDivisionError:
    print("Cannot divide by zero!")
```

3. Finally Block

Code inside finally always runs, no matter if there was an exception or not. Useful for cleanup.

```
try:
    file = open("data.txt", "r")
    data = file.read()
except FileNotFoundError:
```

```
    print("File not found!")
finally:
    print("Closing the file...")
    file.close()
```

90)Write python program that user to enter only odd numbers, else will raise an exception.

Ans.
```
try:
    num = int(input("Enter an odd number: "))
    if num % 2 == 0:
        raise Exception("That’s not an odd number!")
    print("Thanks for entering:", num)
except ValueError:
    print("Please enter a valid number.")
except Exception as e:
    print(e)
```