PYTHON ASSIGNMENT

Q1: What are the types of Applications?

ANS: 1)Desktop Application

2)Mobile Application

3)Web Application

4)Hybrid Application

5)Gaming Application

6)Education Application

Q2: What is programing?

ANS: Programming also known as coding, is the process of designing , writing , testing , and maintaining the source code of computer programs.

Q3: What is Python?

ANS: Python is a high-level, beginning friendly programming language known as its simplicity and versatility.

➔ It was created by Guido van Rossum in 1991.

Q4: Write a Python program to check if a number is positive, negative or zero ?

ANS: num = float(input("enter no:"))

if num>0:

```
    print("The number is positive:")
elif num<0:
    print("The number is negative:")
else :
    print("The number is Zero:")
```

Q5: Write a Python program to get the Factorial number of given numbers ?

ANS:
```
num = int(input("enter the no:"))
factorial = 1
for i in range(1,num+1):
    factorial += i
print(f"factorial {num} is {factorial}")
```

Q6: Write a Python program to get the Fibonacci series of given range ?

ANS:
```
n = int(input("Enter range: "))
a = 0
b = 1

for i in range(n):
    print(a, end=" ")
```

a, b = b, a + b

Q7: How memory is managed in Python?

ANS: Python handles memory management automatically , which is one of the reason it's so beginner friendly.

Q8)What is the purpose continuing statement in python?

ANS: The Continue Statement in python is used to skip the rest of the current iteration in a loop and move on to the next iteration.

Example: For i in range(1,6):

```
        If i ==3:
          Continue
        Print(i)
```

Q9) Write python program that swap two number with temp variable and without temp variable ?

ANS: a = int(input("Enter first number: "))

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

print("\nBefore swapping: a =", a, ", b =", b)

temp = a

```
a = b

b = temp


print("After swapping (with temp): a =", a, ", b =", b)


#  Swap without temp variable

a = a + b

b = a - b

a = a - b


print("After swapping (without temp): a =", a, ", b =", b)
```

Q10) Write a Python program to find whether a given number is even or odd, print out an appropriate message to the user?

ANS:
```
num = int(input("Enter the no:"))

if num % 2 == 0:

    print(f"{num} is even")

else:

    print(f"{num} is odd")
```

Q11) Write a Python program to test whether a passed letter is a vowel or not ?

ANS:
```
letter = input("enter a letter:")
if letter in "aeiouAEIOU":
    print("passed letter is vowel")
else:
    print("passed letter is not vowel")
```

Q12) 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 number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))

if a == b or b == c or a == c:
    total = 0
else:
    total = a + b + c

print("Sum is:", total)
```

Q13) Write a Python program that will return true if the two given integer values are equal or their sum or difference is 5 ?

ANS:
```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))

if a == b or (a + b == 5) or (abs(a - b) == 5):
    print(True)
else:
    print(False)
```

Q14) Write a python program to sum of the first n positive integers ?

ANS:
```
n = int(input("Enter n: "))
total = 0

for i in range(1, n+1):
    total += i

print("Sum is:", total)
```

Q15) Write a Python program to calculate the length of a string ?

ANS: text = input("Enter a string: ")

print("Length of string:", len(text))

Q16) Write a Python program to count the number of characters (character frequency) in a string?

ANS: text = input("Enter a string: ")

freq = {}

for char in text:

    freq[char] = freq.get(char, 0) + 1

print("Character frequency:", freq)

Q17) What are negative indexes and why are they used?

ANS: Negative indexes in Python are used to access elements from the end of a sequence (like a list, string, or tuple).

- -1 refers to the last element.
- -2 refers to the second last element, and so on.

- They are useful when you want elements from the end without calculating the length of the sequence.

Example:
For the list [10, 20, 30, 40]

- list[-1] → 40
- list[-2] → 30

Q18) Write a Python program to count occurrences of a substring in a string ?

ANS: text = input("Enter the main string: ")

sub = input("Enter the substring: ")

count = text.count(sub)

Q19) Write a Python program to count the occurrences of each word in a given sentence ?

ANS: sentence = input("Enter a sentence: ")

words = sentence.split()

for word in set(words):

    print(word, "=", words.count(word))

Q20) 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:
```
str1 = input("Enter first string: ")
str2 = input("Enter second string: ")

new_str1 = str2[:2] + str1[2:]
new_str2 = str1[:2] + str2[2:]

result = new_str1 + " " + new_str2
print("Result:", result)
```

Q21) 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:
```
def add_string(s):
    if len(s) < 3:
        return s
    elif s.endswith('ing'):
        return s + 'ly'
```

```
    else:
        return s + 'in'
print(add_string("play"))
print(add_string("go"))
print(add_string("way"))
```

Q22) Write a Python function to reverses a string if its length is a multiple of 4 ?

ANS:
```
def reverse_if_multiple_of_4(s):
    if len(s) % 4 == 0:
        return s[::-1]
    else:
        return s

# Test
print(reverse_if_multiple_of_4("abcd"))
print(reverse_if_multiple_of_4("abcdef"))
```

Q23) 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:
```
def first_and_last_two(s):
```

```
    if len(s) < 2:
        return ''
    return s[:2] + s[-2:]

# Take input from user
user_input = input("Enter a string: ")

# Call function and print result
result = first_and_last_two(user_input)
print("Result:", result)
```

Q24) Write a Python function to insert a string in the middle of a string ?

ANS:
```
def insert_in_middle(main_str, insert_str):
    middle = len(main_str) // 2  # Find the middle index
    return main_str[:middle] + insert_str + main_str[middle:]

# Take input from user
main_string = input("Enter the main string: ")
insert_string = input("Enter the string to insert: ")
```

```
# Call the function and print result
result = insert_in_middle(main_string, insert_string)
print("Result:", result)
```

Q25) What is List? How will you reverse a list?

- ANS: A list is a built-in data type in Python used to store multiple items in a single variable.
- Lists are ordered, mutable (changeable), and can store different data types together.
- Lists are written inside square brackets **[ ]**, with items separated by commas.
- **How to Reverse a List**
- There are multiple ways:

- **1. Using `reverse()` method (changes the original list)**

```
numbers = [1, 2, 3, 4, 5]
numbers.reverse()
print(numbers)  # [5, 4, 3, 2, 1]
```

- **2. Using slicing (`[::-1]`) (creates a new reversed list)**

```
numbers = [1, 2, 3, 4, 5]
reversed_list = numbers[::-1]
print(reversed_list)  # [5, 4, 3, 2, 1]
```

---

3. Using `reversed()` function

```
numbers = [1, 2, 3, 4, 5]
reversed_list = list(reversed(numbers))
print(reversed_list)  # [ 5, 4, 3, 2, 1]
```

Q26) How will you remove last object from a list?

ANS: **1. Using pop() without an index**

pop() removes and returns the last element by default.

```
my_list = [10, 20, 30, 40]
my_list.pop()
print(my_list)  # [10, 20, 30]
```

---

2. Using slicing ([:-1])

This creates a **new list** without the last element.

```
my_list = [10, 20, 30, 40]
my_list = my_list[:-1]
print(my_list)  # [10, 20, 30]
```

---

3. Using del statement

Deletes the last element in place.

```
my_list = [10, 20, 30, 40]
del my_list[-1]
print(my_list)  # [10, 20, 30]
```

Q27) Suppose list1 is [2, 33, 222, 14, and 25], what is list1 [-1]?

ANS: **If**

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

then

```
list1[-1]
```

means **"the last element of the list"** (because negative indexing in Python starts from the end).

Answer:

```
list1[-1]  # 25
```

So the output will be **25**.

Q28) Differentiate between append () and extend () methods?

ANS: **1. append()**

- **Purpose:** Adds a single element to the end of the list.
- **Effect:** The whole object (item) is added as one element, even if it's another list.
- **Syntax:**

list.append(item)

Example:

my_list = [1, 2, 3]

my_list.append([4, 5]) # Adds the list [4, 5] as a single element

print(my_list)

Output: [1, 2, 3, [4, 5]]

---

2. extend()

- **Purpose:** Adds multiple elements to the list from another iterable (list, tuple, string, etc.).
- **Effect:** Each element from the iterable is added individually.
- **Syntax:**

list.extend(iterable)

Example:

```
my_list = [1, 2, 3]
my_list.extend([4, 5])  # Adds elements 4 and 5 separately
print(my_list)
# Output: [1, 2, 3, 4, 5]
```

Q29) Write a Python function to get the largest number, smallest num and sum of all from a list ?

ANS:

```
def list_stats(numbers):
    largest = max(numbers)       # Largest number
    smallest = min(numbers)      # Smallest number
    total_sum = sum(numbers)     # Sum of all numbers
    return largest, smallest, total_sum

# Take input from user
user_input = input("Enter numbers separated by spaces: ")

# Convert the input string to a list of integers
num_list = list(map(int, user_input.split()))
```

```
# Call the function
largest, smallest, total = list_stats(num_list)

# Print results
print("Largest number:", largest)
print("Smallest number:", smallest)
print("Sum of all numbers:", total)
```

Q30) How will you compare two lists?

ANS: **1. Check if two lists are exactly the same (order + values)**

Use the == operator:

```
list1 = [1, 2, 3]
list2 = [1, 2, 3]
print(list1 == list2)  # True
```

Here, both **elements** and their **order** must match.

---

2. Check if two lists have the same elements (order doesn't matter)

Sort both lists before comparing:

```
list1 = [1, 2, 3]

list2 = [3, 2, 1]

print(sorted(list1) == sorted(list2))  # True
```

3. Check items present in one list but not the other

Use **set operations**:

```
list1 = [1, 2, 3]

list2 = [2, 3, 4]

print(set(list1) - set(list2))  # {1} → in list1 but not in list2

print(set(list2) - set(list1))  # {4} → in list2 but not in list1
```

Q31) 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:

```
# Given list of strings

string_list = ['abc', 'xyz', 'aba', '1221', 'aa', 'bb', 'ccc']


# Counter for strings matching the condition

count = 0
```

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

print("Number of strings with same first and last character:", count)
```

Q32) Write a Python program to remove duplicates from a list ?

ANS:

```
# Given list with duplicates
my_list = [1, 2, 3, 2, 4, 1, 5, 3]

# Removing duplicates while preserving order
unique_list = []
for item in my_list:
    if item not in unique_list:
        unique_list.append(item)

print("List after removing duplicates:", unique_list)
```

Q33) Write a Python program to check a list is empty or not ?

ANS: my_list = []

```
# Check if list is empty
if not my_list:
    print("The list is empty.")
else:
    print("The list is not empty.")
```

Q34) 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

# Example usage:
list_a = [1, 2, 3, 4]
list_b = [5, 6, 3, 8]
```

```
print(common_member(list_a, list_b))
```

Q35) 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:

```
# Generate list of squares from 1 to 30
squares = [x**2 for x in range(1, 31)]

# Print first 5 and last 5 elements
print("First 5 elements:", squares[:5])
print("Last 5 elements:", squares[-5:])
```

Q36) Write a Python function that takes a list and returns a new list with unique elements of the first list?

ANS:

```
def get_unique_elements(lst):
    unique_list = []
    for item in lst:
        if item not in unique_list:
            unique_list.append(item)
    return unique_list
```

```
# Example usage:
my_list = [1, 2, 2, 3, 4, 4, 5]
result = get_unique_elements(my_list)
print(result)
```

Q37) Write a Python program to convert a list of characters into a string?

ANS:
```
chars = ['H', 'e', 'l', 'l', 'o']
string = ''.join(chars)
print("String from characters:",string)
```

Q38) Write a Python program to select an item randomly from a list ?

ANS:
```
import random
items = [1, 2, 3, 4, 5]
random_item = random.choice(items)
print("Randomly selected item:",random_item)
```

Q39) Write a Python program to find the second smallest number in a list ?

ANS:
```
def second_smallest(numbers):
```

```
    unique_numbers = list(set(numbers))  # Remove duplicates
    unique_numbers.sort()                # Sort in ascending order
    return unique_numbers[1]             # Second smallest

# Example usage:
nums = [5, 1, 8, 3, 1, 9]
print("Second smallest number is:", second_smallest(nums))
```

Q40) Write a Python program to get unique values from a list?

ANS:

```
# Using set() to get unique values
def unique_values(lst):
    return list(set(lst))

# Example usage:
numbers = [1, 2, 2, 3, 4, 4, 5]
print("Unique values:", unique_values(numbers))
```

Q41) 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):
    print("Yes, sublist is present")
else:
    print("No, sublist is not present")
```

Q42) Write a Python program to split a list into different variables ?

ANS:
```
# Example list
my_list = [10, 20, 30]

# Split into variables
a, b, c = my_list

# Print results
print(a)
print(b)
```

print(c)

Q43) What is tuple? Difference between list and tuple?

ANS: A **tuple** in Python is a collection of items that:

- Can store **multiple data types** (integers, strings, etc.)
- Is **ordered** (elements have a fixed position)
- Is **immutable** (cannot be changed after creation)
- **Difference between List and Tuple**

Feature	List	Tuple
Syntax	Defined using `[]`	Defined using `()`
Mutability	Mutable – can be changed	Immutable – cannot be changed
Performance	Slightly slower	Faster than lists
Methods	Many built-in methods like `append()`, `remove()`	Fewer built-in methods
Usage	Used when data can change	Used when data should not change

Q44) Write a Python program to create a tuple with different data types ?

ANS: # Tuple with different data types

my_tuple = (25, "Trusha", 3.14, True)

print(my_tuple)

```
print(type(my_tuple))
```

Q45) Write a Python program to unzip a list of tuples into individual lists ?

ANS:

```
# List of tuples
list_of_tuples = [(1, 'a'), (2, 'b'), (3, 'c')]

# Unzipping
numbers, letters = zip(*list_of_tuples)

# Convert to lists
numbers = list(numbers)
letters = list(letters)

print(numbers)
print(letters)
```

Q46) Write a Python program to convert a list of tuples into a dictionary ?

ANS:

```
# List of tuples
list_of_tuples = [(1, 'apple'), (2, 'banana'), (3, 'cherry')]
```

```
# Convert to dictionary
my_dict = dict(list_of_tuples)

# Print result
print(my_dict)
```

Q47) How will you create a dictionary using tuples in python?

ANS:

```
# Tuple of key-value pairs
tuple_data = ((1, 'apple'), (2, 'banana'), (3, 'cherry'))

# Create dictionary
my_dict = dict(tuple_data)

print(my_dict)
```

OR

2. Using a loop

If you want more control (e.g., handle duplicate keys):

```
tuple_data = ((1, 'apple'), (2, 'banana'), (3, 'cherry'))
```

```
my_dict = {}
for key, value in tuple_data:
    my_dict[key] = value

print(my_dict)
```

Q48) Write a Python script to sort (ascending and descending) a dictionary by value?

ANS:

```
my_dict = {'apple': 3, 'banana': 1, 'cherry': 2}

# Ascending order
asc = dict(sorted(my_dict.items(), key=lambda x: x[1]))
print("Ascending:", asc)

# Descending order
desc = dict(sorted(my_dict.items(), key=lambda x: x[1], reverse=True))
print("Descending:", desc)
```

Q49) Write a Python script to concatenate following dictionaries to create a new one ?

ANS: dict1 = {'a': 1, 'b': 2}

dict2 = {'c': 3, 'd': 4}

dict3 = {'e': 5, 'f': 6}

```
# Method 1: Using dictionary unpacking
new_dict = {**dict1, **dict2, **dict3}
print(new_dict)
```

OR

```
#method 2 : update()function
new_dict = {}
for d in (dict1, dict2, dict3):
    new_dict.update(d)
print(new_dict)
```

Q50)Write a Python script to check if a given key already exists in a dictionary?

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

key_to_check = 'b'

```
if key_to_check in my_dict:
    print(f"Key '{key_to_check}' exists in the dictionary.")
else:
    print(f"Key '{key_to_check}' does NOT exist in the dictionary.")
```

Q51) How Do You Traverse Through a Dictionary Object in Python?

ANS: Traversing through a dictionary in Python means going through its keys, values, or key-value pairs one by one. Here are some common ways to do it:

---

1. Traverse through keys only:

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

for key in my_dict:
    print(key)
```

Or explicitly:

```
for key in my_dict.keys():
    print(key)
```

---

2. Traverse through values only:

```
for value in my_dict.values():
    print(value)
```

---

3. Traverse through key-value pairs:

```
for key, value in my_dict.items():
    print(key, value)
```

---

4. Traverse with index (if needed):

If you want to get an index along with items, you can use enumerate():

```
for index, (key, value) in enumerate(my_dict.items()):
    print(index, key, value)
```

Q52)How Do You Check the Presence of a Key in A Dictionary?

ANS: **1. Using the in keyword**

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

if 'b' in my_dict:
    print("Key 'b' is present.")
```

```
else:
    print("Key 'b' is not present.")
```

2. Using the get() method

The get() method returns None (or a default value you specify) if the key is not found.

```
if my_dict.get('b') is not None:
    print("Key 'b' is present.")
else:
    print("Key 'b' is not present.")
```

3. Using dict.keys()

You can check if the key is in the list of keys.

```
if 'b' in my_dict.keys():
    print("Key 'b' is present.")
else:
    print("Key 'b' is not present.")
```

Q53) 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)
```

Q54) Write a Python program to check multiple keys exists in a dictionary ?

ANS:

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

keys_to_check = ['a', 'b', 'd']

all_exist = True
for key in keys_to_check:
    if key not in my_dict:
        all_exist = False
        break

if all_exist:
    print("All keys are present.")
else:
    print("Some keys are missing.")
```

Q55) Write a Python script to merge two Python dictionaries ?

ANS: dict1 = {'a': 1, 'b': 2}

dict2 = {'c': 3, 'd': 4}

dict1.update(dict2) # Adds items from dict2 into dict1

print(dict1)

Q56) Write a Python program to map two lists into a dictionary

Sample output: Counter ({'a': 400, 'b': 400,'d': 400, 'c': 300}).

ANS: keys = ['a', 'b', 'c', 'd']

values = [400, 400, 300, 400]

Combine lists into a dictionary

result = dict(zip(keys, values))

print(result)

Q57)Write a Python program to find the highest 3 values in a dictionary?

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

```
# Sort values in descending order and get first 3
top3 = sorted(my_dict.values(), reverse=True)[:3]

print(top3)
```

Q58) 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 d in data:
    if d['item'] in result:
        result[d['item']] += d['amount']
    else:
        result[d['item']] = d['amount']

print(result)
```

Q59)Write a Python program to create a dictionary from a string. Note: Track the count of the letters from the string ?

ANS:

```
string = "hello world"

count_dict = {}

for char in string:
    if char.isalpha():  # count only letters, ignore spaces or punctuation
        if char in count_dict:
            count_dict[char] += 1
```

```
        else:
            count_dict[char] = 1

print(count_dict)
```

Q60) Sample string: 'w3resource'

Expected output: • {'3': 1,’s’: 1, 'r': 2, 'u': 1, 'w': 1, 'c': 1, 'e': 2, 'o': 1}

ANS:

```
sample_str = 'w3resource'

char_count = {}
for char in sample_str:
    char_count[char] = char_count.get(char, 0) + 1

print(char_count)
```

Q61) Write a Python function to calculate the factorial of a number (a nonnegative integer)?

ANS:

```
def factorial(n):
    if n == 0:
        return 1
```

```
    else:
        return n * factorial(n - 1)
print(factorial(9))
```

Q62) Write a Python function to check whether a number is in a given range?

ANS:
```
def check_number(num, start, end):
    if num >= start and num <= end:
        return True
    else:
        return False

# Example:
print(check_number(7, 1, 10))
print(check_number(12, 1, 10))
```

Q63) Write a Python function to check whether a number is perfect or not?

ANS:
```
def is_perfect(num):
    sum_div = 0
    for i in range(1, num):
```

```
        if num % i == 0:
            sum_div += i
    if sum_div == num:
        return True
    else:
        return False

# Examples:
print(is_perfect(6))   # True
print(is_perfect(10))
```

Q64) Write a Python function that checks whether a passed string is palindrome or not?

ANS:
```
def is_palindrome(s):
    return s == s[::-1]

# Example:
print(is_palindrome("madam"))
print(is_palindrome("hello"))
```

Q65) How Many Basic Types of Functions Are Available in Python?

ANS: In Python, there are mainly **4 basic types of functions**:

1. **Built-in functions**
   Functions that come with Python, like print(), len(), type(), etc.
2. **User-defined functions**
   Functions you create yourself using the def keyword.
3. **Anonymous functions (Lambda functions)**
   Small, unnamed functions created with the lambda keyword.
4. **Recursive functions**
   Functions that call themselves to solve problems.

Q66) How can you pick a random item from a list or tuple?

ANS: import random

```
items = [1, 2, 3, 4, 5]  # This is a list
print(random.choice(items))  # Picks a random item from the list
```

```
fruits = ('apple', 'banana', 'cherry')  # This is a tuple
print(random.choice(fruits))
```

Q67) 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)
```

Q68) How can you get a random number in python?

ANS: import random

```
# Get a random number between 1 and 10
number = random.randint(1, 10)
```

print(number)

Q69) How will you set the starting value in generating random numbers?

ANS: import random

random.seed(10) # Set the starting value (seed) to 10

print(random.randint(1, 100)) # This will always print the same number when seed is 10

print(random.random())

Q70) How will you randomize the items of a list in place?

ANS: import random

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

random.shuffle(my_list)

print(my_list)

Q71) What is File function in python? What are keywords to create and write file?

ANS: **What is the file function in Python**

Actually, in modern Python, there is no separate file() function. Instead, Python uses the built-in function open() to create, open, read, write, and work with files.

How to create and write to a file in Python

You use the open() function with the right mode (keyword) to create and write files.

Common keywords (modes) to create and write files:

Mode	Meaning
'w'	Write mode: create a file or overwrite if exists
'a'	Append mode: add data to the end if file exists
'x'	Create mode: create a new file, fail if file exists

Q 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)
```

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

ANS:

```
# Open file in append mode
file = open('filename.txt', 'a')

# Add new text
file.write("This is new text.\n")

# Close the file
file.close()

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

```
# Read and print all content
print(file.read())

# Close the file
file.close()
```

Q74) Write a Python program to read first n lines of a file?

ANS:
```
filename = "example.txt"  # your file name
n = 5  # number of lines to read

file = open(filename, "r")

for i in range(n):
    line = file.readline()
    if not line:
        break
    print(line, end="")

file.close()
```

Q75) Write a Python program to read last n lines of a file?

ANS:
```
filename = "example.txt"  # your file name
n = 5  # number of lines to read from the end

with open(filename, "r") as file:
    lines = file.readlines()  # read all lines into a list

last_lines = lines[-n:]  # get last n lines

for line in last_lines:
    print(line, end="")
```

Q76) Write a Python program to read a file line by line and store it into a list?

ANS:
```
# Specify the filename
filename = 'yourfile.txt'

# Initialize an empty list to store lines
lines = []
```

```
# Open the file in read mode
with open(filename, 'r') as file:
    for line in file:
        lines.append(line.rstrip('\n'))  # Remove newline character and add to list

# Print the list of lines
print(lines)
```

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

ANS: filename = 'yourfile.txt'

```
# Initialize an empty string to store the contents
content = ''

with open(filename, 'r') as file:
    for line in file:
        content += line  # Keep the newline characters as they are
```

```
print(content)
```

Q78) 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)
```

Q79) 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)
```

Q80) 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)
```

Q81) 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.")
```

Q82) 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!")
```

Q83) 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.

Q84) How many except statements can a try-except block have? Name Some built-in exception classes ?

ANS: **How many except statements can you use?**

- 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.

Q85) 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.

Q86) 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.

Q87) 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.

Q88) 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**.

Q89) 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()
```

Q90) 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)
```