**Class XII**

**Computer Science**

**Short Questions & Answers**

**Topic: Functions in Python**

**Short Questions & Answers**

1. What is a text file? Answer: A text file is a file that contains only plain text, without any formatting or binary data.
2. What is a binary file? Answer: A binary file is a file that contains data in binary format, which is not human-readable.
3. What is a CSV file? Answer: A CSV file is a file format used to store tabular data in a plain text form.
4. What is a relative path? Answer: A relative path is a path that specifies the location of a file relative to the current directory.
5. What is an absolute path? Answer: An absolute path is a path that specifies the complete path to a file, starting from the root directory.
6. How do you open a text file in Python? Answer: You can open a text file in Python using the open() function.
7. What are the different text file open modes in Python? Answer: The different text file open modes in Python are r, r+, w, w+, a, and a+.
8. How do you close a text file in Python? Answer: You can close a text file in Python using the close() method.
9. What is the with clause in Python? Answer: The with clause is a Python statement that provides a context for the execution of a block of code.
10. How do you write data to a text file in Python? Answer: You can write data to a text file in Python using the write() and writelines() methods.
11. How do you append data to a text file in Python? Answer: You can append data to a text file in Python using the append mode (a or a+) while opening the file.
12. How do you read data from a text file in Python? Answer: You can read data from a text file in Python using the read(), readline(), and readlines() methods.
13. What is the seek method in Python? Answer: The seek method in Python is used to change the position of the file pointer within a file.
14. What is the tell method in Python? Answer: The tell method in Python is used to return the current position of the file pointer within a file.
15. How do you manipulate data in a text file in Python? Answer: You can manipulate data in a text file in Python by reading the data into memory, making changes to it, and then writing the modified data back to the file.
16. What are some common operations performed on text files? Answer: Some common operations performed on text files include reading, writing, appending, and modifying data.
17. What is the difference between a text file and a binary file? Answer: A text file contains human-readable text, while a binary file contains data in a non-human-readable format.
18. What are the advantages of using relative paths in file handling? Answer: Relative paths are easier to use and are more portable across different platforms.
19. What are the advantages of using absolute paths in file handling? Answer: Absolute paths provide a complete and unambiguous specification of the location of a file.
20. What is file handling in computer science? Answer: File handling refers to the process of working with files, including reading, writing, and manipulating file data.
21. What is a file pointer in Python? Answer: A file pointer is a variable that points to a specific position within a file, allowing for the reading and writing of data.
22. How do you open a binary file in Python? Answer: You can open a binary file in Python using the open() function with the "b" flag.
23. What is a delimiter in a CSV file? Answer: A delimiter is a character or set of characters used to separate data fields within a CSV file.
24. How do you write data to a CSV file in Python? Answer: You can write data to a CSV file in Python using the csv module and the writerow() method.
25. How do you read data from a CSV file in Python? Answer: You can read data from a CSV file in Python using the csv module and the reader() method.

Programs:

1. Write a program to open a text file and count the number of characters, words, and lines in it.

with open('textfile.txt', 'r') as file:

 contents = file.read()

 char\_count = len(contents)

 word\_count = len(contents.split())

 line\_count = len(contents.split('\n'))

 print(f"Number of characters: {char\_count}")

 print(f"Number of words: {word\_count}")

 print(f"Number of lines: {line\_count}")

1. Write a program to open a binary file and convert the data into a readable format.

with open('binaryfile.bin', 'rb') as file:

 binary\_data = file.read()

 readable\_data = binary\_data.decode('utf-8')

 print(readable\_data)

1. Write a program to read a CSV file and find the sum of a particular column.

import csv

with open('csvfile.csv', 'r') as file:

 reader = csv.reader(file)

 column\_sum = 0

 for row in reader:

 column\_sum += int(row[2])

 print(f"Sum of column 3: {column\_sum}")

1. Write a program to read a text file, replace a word with another word, and save the modified file.

with open('textfile.txt', 'r') as file:

 contents = file.read()

 modified\_contents = contents.replace('old\_word', 'new\_word')

with open('textfile.txt', 'w') as file:

 file.write(modified\_contents)

1. Write a program to read a binary file and find the number of occurrences of a particular byte.

with open('binaryfile.bin', 'rb') as file:

 binary\_data = file.read()

 byte\_count = binary\_data.count(b'\x01')

 print(f"Number of occurrences of byte \\x01: {byte\_count}")

1. Write a program to read a CSV file and sort the rows based on a particular column.

import csv

with open('csvfile.csv', 'r') as file:

 reader = csv.reader(file)

 sorted\_rows = sorted(reader, key=lambda row: row[2])

 for row in sorted\_rows:

 print(row)

1. Write a program to read a text file and create a dictionary that contains the frequency of each word.

with open('textfile.txt', 'r') as file:

 contents = file.read()

 words = contents.split()

 word\_count = {}

 for word in words:

 if word not in word\_count:

 word\_count[word] = 1

 else:

 word\_count[word] += 1

 print(word\_count)

1. Write a program to read a binary file and find the maximum and minimum values.

import struct

with open('binaryfile.bin', 'rb') as file:

 binary\_data = file.read()

 values = struct.unpack('f'\*int(len(binary\_data)/4), binary\_data)

 max\_value = max(values)

 min\_value = min(values)

 print(f"Maximum value: {max\_value}")

 print(f"Minimum value: {min\_value}")

1. Write a program to read a CSV file and find the average of a particular column.

import csv

with open('csvfile.csv', 'r') as file:

 reader = csv.reader(file)

 column\_sum = 0

 row\_count = 0

 for row in reader:

 column\_sum += int(row[2])

 row\_count += 1

 column\_average =

1. Write a program to read a text file and create a new file that contains only the lines that contain a specific keyword.

with open('textfile.txt', 'r') as file:

 contents = file.readlines()

 keyword = 'specific\_keyword'

 filtered\_lines = [line for line in contents if keyword in line]

with open('filtered\_file.txt', 'w') as file:

 file.writelines(filtered\_lines)

**Practice Questions**

1. Write a program that calculates the factorial of a given number using a user-defined function.
2. Write a program that determines whether a given number is a prime number using a user-defined function.
3. Write a program that finds the greatest common divisor (GCD) of two numbers using a user-defined function.
4. Write a program that calculates the sum of a series of numbers (e.g., 1+2+3+...+n) using a user-defined function.
5. Create a user-defined function that returns the length of a given string without using the built-in **len()** function.
6. Write a program that accepts a list of numbers and returns a new list containing only the even numbers using a user-defined function.
7. Create a user-defined function that accepts two strings and returns a concatenated string without using the **+** operator.
8. Write a program that calculates the sum of the diagonal elements of a square matrix using a user-defined function.
9. Write a program that accepts a list of numbers and returns the average of those numbers using a user-defined function.
10. Create a user-defined function that accepts a list of strings and returns the longest string in the list.
11. Write a program that sorts a list of numbers in ascending order using a user-defined function.
12. Write a program that reverses a given string without using the built-in **reverse()** function or string slicing.
13. Create a user-defined function that checks if a given word is a palindrome (reads the same forward and backward).
14. Write a program that calculates the Fibonacci sequence up to a given number using a user-defined function.
15. Create a user-defined function that accepts a list of numbers and returns the maximum and minimum values in the list.
16. Write a program that converts a given temperature from Celsius to Fahrenheit and vice versa using user-defined functions.
17. Write a program that finds the common elements between two given lists using a user-defined function.
18. Create a user-defined function that calculates the area and circumference of a circle, given its radius.
19. Write a program that counts the occurrences of a specific character in a given string using a user-defined function.
20. Write a program that accepts a list of numbers and returns a list of unique elements from the original list using a user-defined function.