|  |
| --- |
| Question 1: |
|  |

Define a class with a generator which can iterate the numbers, which are divisible by 7, between a given range 0 and n.

**Ans:- class DivisibleBySeven:**

**def \_\_init\_\_(self, n):**

**self.n = n**

**def divisible\_by\_seven\_generator(self):**

**for num in range(self.n + 1):**

**if num % 7 == 0:**

**yield num**

**# Example usage**

**n = 50**

**divisible\_by\_seven = DivisibleBySeven(n)**

**generator = divisible\_by\_seven.divisible\_by\_seven\_generator()**

**for number in generator:**

**print(number)**

Question 2:

|  |
| --- |
| Write a program to compute the frequency of the words from the input. The output should output after sorting the key alphanumerically. |
|  |

|  |
| --- |
| Suppose the following input is supplied to the program: |
|  |

|  |
| --- |
| New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3. |
|  |

|  |
| --- |
| Then, the output should be: |
|  |

|  |
| --- |
| 2:2 |
|  |

|  |
| --- |
| 3.:1 |
|  |

|  |
| --- |
| 3?:1 |
|  |

|  |
| --- |
| New:1 |
|  |

|  |
| --- |
| Python:5 |
|  |

|  |
| --- |
| Read:1 |
|  |

|  |
| --- |
| and:1 |
|  |

|  |
| --- |
| between:1 |
|  |

|  |
| --- |
| choosing:1 |
|  |

|  |
| --- |
| or:2 |
|  |

to:1

**Ans:- def frequency(a):**

**result = {}**

**result\_1 = a.split()**

**for i in result\_1:**

**if i not in result:**

**result[i] = 1**

**else:**

**result[i] += 1**

**# Sort the keys alphabetically**

**sorted\_keys = sorted(result.keys())**

**# Print the word frequencies**

**for key in sorted\_keys:**

**print(key, ":", result[key])**

**frequency("New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3.")**

|  |
| --- |
| Question 3: |
|  |

|  |
| --- |
|  |
|  |

Define a class Person and its two child classes: Male and Female. All classes have a method "getGender" which can print "Male" for Male class and "Female" for Female class.

**Ans:- class Person:**

**def getGender(self):**

**print("Unknown")**

**class Male(Person):**

**def getGender(self):**

**print("Male")**

**class Female(Person):**

**def getGender(self):**

**print("Female")**

**# Example usage**

**person = Person()**

**person.getGender() # Output: Unknown**

**male = Male()**

**male.getGender() # Output: Male**

**female = Female()**

**female.getGender() # Output: Female**

Question 4:

Please write a program to generate all sentences where subject is in ["I", "You"] and verb is in ["Play", "Love"] and the object is in ["Hockey","Football"].

**Ans:- subjects = ["I", "You"]**

**verbs = ["Play", "Love"]**

**objects = ["Hockey", "Football"]**

**def generate\_sentences():**

**sentences = []**

**for subject in subjects:**

**for verb in verbs:**

**for obj in objects:**

**sentence = f"{subject} {verb} {obj}."**

**sentences.append(sentence)**

**return sentences**

**# Example usage**

**sentences = generate\_sentences()**

**for sentence in sentences:**

**print(sentence)**

Question 5:

Please write a program to compress and decompress the string "hello world!hello world!hello world!hello world!".

**Ans:- import zlib**

**def compress\_string(string):**

**# Convert the string to bytes**

**input\_bytes = string.encode('utf-8')**

**# Compress the bytes**

**compressed\_bytes = zlib.compress(input\_bytes)**

**return compressed\_bytes**

**def decompress\_string(compressed\_bytes):**

**# Decompress the bytes**

**decompressed\_bytes = zlib.decompress(compressed\_bytes)**

**# Convert the decompressed bytes to a string**

**decompressed\_string = decompressed\_bytes.decode('utf-8')**

**return decompressed\_string**

**# Original string**

**original\_string = "hello world!hello world!hello world!hello world!"**

**print("Original:", original\_string)**

**# Compress the string**

**compressed\_bytes = compress\_string(original\_string)**

**print("Compressed:", compressed\_bytes)**

**# Decompress the string**

**decompressed\_string = decompress\_string(compressed\_bytes)**

**print("Decompressed:", decompressed\_string)**

Question 6:

Please write a binary search function which searches an item in a sorted list. The function should return the index of element to be searched in the list.

**Ans:- def binary\_search(target, lst):**

**left = 0**

**right = len(lst) - 1**

**while left <= right:**

**mid = (left + right) // 2**

**if lst[mid] == target:**

**return mid**

**elif lst[mid] < target:**

**left = mid + 1**

**else:**

**right = mid - 1**

**# Element not found**

**return -1**

**# Example usage**

**sorted\_list = ["kapish", "ishan", "manya", "mansi", "sharwari"]**

**target = "sharwari"**

**result = binary\_search(target, sorted\_list)**

**if result != -1:**

**print(f"'{target}' found at index: {result}")**

**else:**

**print(f"'{target}' not found in the list.")**