|  |
| --- |
| **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.**

class DivisibleBySeven:

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

self.n = n

def divisible\_by\_seven(self):

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

if num % 7 == 0:

yield num

# Create an instance of the class

divisible\_by\_seven\_obj = DivisibleBySeven(50)

# Iterate over the numbers divisible by 7 using the generator

for number in divisible\_by\_seven\_obj.divisible\_by\_seven():

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**

def word\_frequency(input\_text):

word\_count = {}

# Split the input text into words

words = input\_text.split()

# Count the frequency of each word

for word in words:

if word in word\_count:

word\_count[word] += 1

else:

word\_count[word] = 1

# Sort the keys alphanumerically

sorted\_keys = sorted(word\_count.keys())

# Print the word frequencies

for key in sorted\_keys:

print(key + ":" + str(word\_count[key]))

# Test the function with the example input

input\_text = "New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3."

word\_frequency(input\_text)

|  |
| --- |
| **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.**

class Person:

def getGender(self):

pass

class Male(Person):

def getGender(self):

print("Male")

class Female(Person):

def getGender(self):

print("Female")

# Create instances of the child classes

male = Male()

female = Female()

# Call the getGender method on the instances

male.getGender()

female.getGender()

**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"].**

subjects = ["I", "You"]

verbs = ["Play", "Love"]

objects = ["Hockey", "Football"]

sentences = []

for subject in subjects:

for verb in verbs:

for obj in objects:

sentence = subject + " " + verb + " " + obj

sentences.append(sentence)

# Print the generated 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!".**

import zlib

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

# Compress the string

compressed\_data = zlib.compress(original\_string.encode())

# Decompress the string

decompressed\_data = zlib.decompress(compressed\_data)

# Convert the decompressed data back to string

decompressed\_string = decompressed\_data.decode()

# Print the results

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

print("Compressed Data:", compressed\_data)

print("Decompressed String:", 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.**

def binary\_search(arr, target):

low = 0

high = len(arr) - 1

while low <= high:

mid = (low + high) // 2

if arr[mid] == target:

return mid

elif arr[mid] < target:

low = mid + 1

else:

high = mid - 1

return -1

# Test the function

sorted\_list = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

target\_element = 23

index = binary\_search(sorted\_list, target\_element)

if index != -1:

print(f"Element {target\_element} found at index {index}")

else:

print("Element not found")