|  |
| --- |
| 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 i in range(self.n + 1):

if i % 7 == 0:

yield i

# Example usage

n = 50

numbers = DivisibleBySeven(n).divisible\_by\_seven()

for num in numbers:

print(num)

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\_string):

word\_dict = {}

words = input\_string.split()

for word in words:

word\_dict[word] = word\_dict.get(word, 0) + 1

sorted\_words = sorted(word\_dict.items())

for word, frequency in sorted\_words:

print(f"{word}:{frequency}")

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

word\_frequency(input\_string)

|  |
| --- |
| 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")

# Example usage

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"].

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 all 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 gzip

def compress\_string(string):

compressed\_data = gzip.compress(string.encode('utf-8'))

return compressed\_data

def decompress\_string(compressed\_data):

decompressed\_data = gzip.decompress(compressed\_data).decode('utf-8')

return decompressed\_data

# Original string

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

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

# Compress the string

compressed\_data = compress\_string(original\_string)

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

# Decompress the string

decompressed\_string = decompress\_string(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):

left = 0

right = len(arr) - 1

while left <= right:

mid = (left + right) // 2

if arr[mid] == target:

return mid

elif arr[mid] < target:

left = mid + 1

else:

right = mid - 1

# If the target is not found in the list

return -1