|  |
| --- |
| 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 generator(self):

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

if i % 7 == 0:

yield i

num = int(input("Enter number : "))

# create an instance of the DivisibleBySeven class

divisible\_by\_seven = DivisibleBySeven(num)

# get the generator object

gen = divisible\_by\_seven.generator()

# iterate over the generator object

for num in gen:

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

input\_string = input()

words = input\_string.split()

# create a dictionary to store the word frequencies

word\_freq = {}

# iterate over the words

for word in words:

# if the word is already in the dictionary, increment its count

if word in word\_freq:

word\_freq[word] = word\_freq[word] + 1

# otherwise, add the word to the dictionary with a count of 1

else:

word\_freq[word] = 1

# sort the dictionary keys alphanumerically

sorted\_words = sorted(word\_freq.keys())

# print the word frequencies

for word in sorted\_words:

print(word, word\_freq[word])

|  |
| --- |
| 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):

return "Male"

class Female(Person):

def getGender(self):

return "Female"

m = Male()

print(m.getGender()) # prints "Male"

f = Female()

print(f.getGender()) # prints "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"]

for subject in subjects:

for verb in verbs:

for object in objects:

print(f"{subject} {verb} {object}.")

Question 5:

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

import zlib

# input srting

string = input("Enter string for compress and decompress : ")

# Compress the string using zlib

compressed = zlib.compress(string.encode())

# Decompress the compressed string

decompressed = zlib.decompress(compressed).decode()

# Print the original, compressed, and decompressed strings

print(f"Original string: {string}")

print(f"Compressed string: {compressed}")

print(f"Decompressed string: {decompressed}")

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(lst, item):

# Set the initial indices for the search

low = 0

high = len(lst) - 1

# Keep searching until the indices meet

while low <= high:

# Calculate the midpoint of the list

mid = (low + high) // 2

# Get the value at the midpoint

guess = lst[mid]

# If the guess is the item, return the index

if guess == item:

return mid

# If the guess is greater than the item, search the left half of the list

if guess > item:

high = mid - 1

# If the guess is less than the item, search the right half of the list

else:

low = mid + 1

# If the item is not found, return -1

return -1

# Create a sorted list

lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Search for the item 5 in the list

item = 3

index = binary\_search(lst, item)

# Print the result

print(f"The item {item} is at index {index} in the list.")