|  |
| --- |
| 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: You can define a class with a generator method that iterates over numbers divisible by 7 within a given range (0 to n). Here's the Python code for the class:

class DivisibleBySevenGenerator:

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

self.n = n

def is\_divisible\_by\_seven(self, num):

return num % 7 == 0

def generate\_numbers(self):

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

if self.is\_divisible\_by\_seven(num):

yield num

# Test the class and generator

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input("Enter the range (0 to n): "))

generator = DivisibleBySevenGenerator(n)

print("Numbers divisible by 7 within the range (0 to {}):".format(n))

for num in generator.generate\_numbers():

print(num)

In this program, we define the `DivisibleBySevenGenerator` class with three methods:

i). `\_\_init\_\_`: The constructor initializes the class with the given range value `n`.

ii). `is\_divisible\_by\_seven`: This method checks if a number is divisible by 7.

iii). `generate\_numbers`: This is the generator method. It iterates over the numbers from 0 to `n` and yields only the numbers that are divisible by 7.The program prompts the user to enter the range value `n`, and then it uses the generator to print all the numbers within the range that are divisible by 7.

Example usage:

Enter the range (0 to n): 50

Numbers divisible by 7 within the range (0 to 50):

0

7

14

21

28

35

42

49

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 compute\_word\_frequency(input\_text):

word\_freq = {}

words = input\_text.split()

for word in words:

word = word.strip('.,?!')

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

return word\_freq

# Test the program

if \_\_name\_\_ == "\_\_main\_\_":

input\_text = input("Enter the input text: ")

word\_frequency = compute\_word\_frequency(input\_text)

# Sort the keys alphanumerically and print the result

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

for key in sorted\_keys:

print(f"{key}:{word\_frequency[key]}")

Example usage:

Enter the input text: New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3.

2:2

3:1

New:1

Python:5

Read:1

and:1

between:1

choosing:1

or:2

to:1

|  |
| --- |
| 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 Gender")

class Male(Person):

def getGender(self):

print("Male")

class Female(Person):

def getGender(self):

print("Female")

# Test the classes

if \_\_name\_\_ == "\_\_main\_\_":

person = Person()

male = Male()

female = Female()

person.getGender() # Output: Unknown Gender

male.getGender() # Output: Male

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: You can create a Python program that generates all possible sentences where the subject is in ["I", "You"], the verb is in ["Play", "Love"], and the object is in ["Hockey", "Football"]. Here's the program using nested loops:

subjects = ["I", "You"]

verbs = ["Play", "Love"]

objects = ["Hockey", "Football"]

sentences = []

for subject in subjects:

for verb in verbs:

for obj in objects:

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

sentences.append(sentence)

# Print the generated sentences

for sentence in sentences:

print(sentence)

In this program, we define lists for subjects, verbs, and objects. Then, we use three nested loops to generate all possible combinations of subject, verb, and object. For each combination, we construct a sentence and add it to the `sentences` list.

Finally, we print all the generated sentences.

Example output:

I Play Hockey.

I Play Football.

I Love Hockey.

I Love Football.

You Play Hockey.

You Play Football.

You Love Hockey.

You Love Football.

Question 5:

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

ANS: To compress and decompress a string, you can use the `gzip` module in Python. The `gzip` module provides functionality to compress and decompress data using the gzip data format. Here's a program to compress and then decompress the string "hello world!hello world!hello world!hello world!":

import gzip

import io

def compress\_string(input\_string):

out = io.BytesIO()

with gzip.GzipFile(fileobj=out, mode="wb") as f:

f.write(input\_string.encode())

compressed\_data = out.getvalue()

return compressed\_data

def decompress\_string(compressed\_data):

in\_data = io.BytesIO(compressed\_data)

with gzip.GzipFile(fileobj=in\_data, mode="rb") as f:

decompressed\_string = f.read().decode()

return decompressed\_string

# Test the program

if \_\_name\_\_ == "\_\_main\_\_":

input\_string = "hello world!hello world!hello world!hello world!"

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

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

decompressed\_string = decompress\_string(compressed\_data)

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

In this program, we define two functions:

A). `compress\_string`: This function takes an input string, compresses it using the gzip format, and returns the compressed data.

B). `decompress\_string`: This function takes compressed data, decompresses it using the gzip format, and returns the original string.

The program compresses the input string using the `compress\_string` function, prints the compressed data, then decompresses the compressed data using the `decompress\_string` function, and prints the decompressed string.

Example output:

Compressed data: b'\x1f\x8b\x08\x00\xbb-\xc6RH\x00\xf3x\x01\x0cH\xe3K\x0c\x08\xea\xffhello world!hello world!hello world!hello world!\x00=\x9fF\x8a'

Decompressed string: hello world!hello world!hello world!hello world!

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(sorted\_list, item):

low = 0

high = len(sorted\_list) - 1

while low <= high:

mid = (low + high) // 2

mid\_item = sorted\_list[mid]

if mid\_item == item:

return mid

elif mid\_item < item:

low = mid + 1

else:

high = mid - 1

return -1

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

sorted\_list = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

item\_to\_search = 9

index = binary\_search(sorted\_list, item\_to\_search)

if index != -1:

print(f"Item found at index: {index}")

else:

print("Item not found in the list.")

Example usage:

Item found at index: 4