|  |
| --- |
| 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.

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

|  |
| --- |
| 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.

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"].

Question 5:

Please write a program to compress and decompress the 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:

Answer 1:

Here is an example of a class with a generator that iterates over numbers divisible by 7 within a given range:

```python

class DivisibleBySeven:

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

self.n = n

def generate(self):

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

if num % 7 == 0:

yield num

# Example usage:

divisible\_by\_seven = DivisibleBySeven(50)

generator = divisible\_by\_seven.generate()

for num in generator:

print(num)

```

In this example, the `DivisibleBySeven` class takes a parameter `n` which represents the upper bound of the range (inclusive). The `generate` method is a generator function that yields numbers divisible by 7 within the given range. You can use the generator by creating an instance of the class and calling the `generate` method, as shown in the example usage.

Answer 2:

Here's a program that computes the frequency of words from the input and sorts the results alphabetically:

```python

from collections import defaultdict

def word\_frequency(text):

frequency = defaultdict(int)

words = text.split()

for word in words:

frequency[word] += 1

sorted\_frequency = sorted(frequency.items(), key=lambda x: x[0])

return sorted\_frequency

# Example usage:

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

result = word\_frequency(input\_text)

for word, count in result:

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

```

The `word\_frequency` function takes the input text as a parameter and returns a list of tuples representing the word frequencies. Each tuple contains a word and its corresponding count. The function uses a `defaultdict` to simplify the counting process. The result is then sorted alphabetically using the `sorted` function with a lambda function as the key.

Answer 3:

Here's an implementation of the `Person`, `Male`, and `Female` classes with a `getGender` method:

```python

class Person:

def getGender(self):

print("Unknown gender")

class Male(Person):

def getGender(self):

print("Male")

class Female(Person):

def getGender(self):

print("Female")

# Example usage:

person = Person()

person.getGender() # Output: Unknown gender

male = Male()

male.getGender() # Output: Male

female = Female()

female.getGender() # Output: Female

```

The `Person` class serves as the base class, and the `Male` and `Female` classes inherit from it. Each class overrides the `getGender` method to provide the appropriate output based on the gender. The example usage demonstrates calling the `getGender` method on instances of each class.

Answer 4:

Here's a program that generates sentences with subjects from ["I", "You"], verbs from ["Play", "Love"], and objects from ["Hockey", "Football"]:

```python

subjects = ["I", "You"]

verbs = ["Play", "Love"]

objects = ["Hockey", "Football"]

sentences = [(subject, verb, obj) for subject in subjects for verb in verbs for obj in objects]

for sentence in sentences:

print(" ".join(sentence))

```

In this program, a list comprehension is used to generate all possible combinations of subject, verb, and object. The result is a list of tuples, where each tuple represents a sentence. The sentences are then printed by joining the words with spaces using `" ".join()`.

Answer 5:

To compress and decompress a string, you can use the `gzip` module in Python. Here's an example:

```python

import gzip

def compress\_string(text):

compressed = gzip.compress(text.encode("utf-8"))

return compressed

def decompress\_string(compressed):

decompressed = gzip.decompress(compressed).decode("utf-8")

return decompressed

# Example usage:

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

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

print(f"Compressed: {compressed\_string}")

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

print(f"Decompressed: {decompressed\_string}")

```

In this example, the `compress\_string` function takes a string and compresses it using `gzip.compress`, which returns a compressed byte string. The `decompress\_string` function takes a compressed byte string and decompresses it using `gzip.decompress`, returning the original string. The strings are encoded and decoded using UTF-8 to handle non-ASCII characters.

Answer 6:

Here's an implementation of a binary search function in Python:

```python

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

# Example usage:

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

target = 7

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

print(f"Index of {target}: {result}")

```

The `binary\_search` function takes a sorted list (`arr`) and a target element (`target`). It initializes `low` and `high` variables representing the lower and upper bounds of the search range. The function iteratively narrows down the search range by updating `low` and `high` based on comparisons with the target value. If the target is found, the index is returned. If the target is not present in the list, `-1` is returned. The example usage demonstrates searching for the target value in a sorted list.