|  |
| --- |
| 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.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

class get\_values():

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

self.number = number

def generate(self):

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

if i%7 == 0:

yield i

try:

seven = get\_values(100)

for i in seven.generate():

print(i, end=" ,")

lg.info("class get\_values has been called")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

test\_string1 = "New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3."

def test\_of\_string(test\_string):

sentence = test\_string.split()

sentence.sort()

print({i:sentence.count(i) for i in sentence})

try:

test\_of\_string(test\_string1)

lg.info("Function has been called")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

|  |
| --- |
| 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 \_\_init\_\_(self, fname, lname):

self.firstname = fname

self.lastname = lname

def printname(self):

print(self.firstname, self.lastname)

class Male(Person):

def \_\_init\_\_(self, fname, lname):

self.fname = fname

self.lname = lname

def getGender(self):

print("Gender is Male")

class Female(Person):

def \_\_init\_\_(self, fname1, lname1):

self.fname1 = fname1

self.lname1 = lname1

def getGender(self):

print("Gender is 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.

subject = ["I", "You"]

verb = ["Play", "Love"]

object1 = ["Hockey","Football"]

for i in subject:

for j in verb:

for k in object1:

my\_sentence = ""

my\_sentence += i+" "+j+" "+k

print(my\_sentence)

Question 5:

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

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

class comp\_and\_decomp():

def \_\_init\_\_(self,my\_string):

self.my\_string = my\_string

def compress(self):

empty\_string = ""

for i in self.my\_string:

empty\_string += i

if empty\_string\*2 in self.my\_string:

break

else:

continue

print(empty\_string)

def decompress(self):

print(self.my\_string)

try:

p = comp\_and\_decomp("hello world!hello world!hello world!hello world!")

p.compress()

p.decompress()

lg.info("Class comp\_and\_decomp() has been called")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def binary\_search(super\_list, search\_element):

super\_list.sort()

main\_list = super\_list.copy()

while len(main\_list) >2:

if search\_element not in main\_list:

print("search element not in list")

break

else:

if main\_list[len(main\_list)//2] < search\_element:

main\_list = main\_list[len(main\_list)//2:len(main\_list)+1]

print(main\_list)

else:

main\_list = main\_list[0:len(main\_list)//2+1]

print(main\_list)

print(main\_list)

print("The position of the element",search\_element,"is",super\_list.index(max(main\_list)))

try:

super\_list = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

binary\_search(super\_list, 6)

lg.info("Class comp\_and\_decomp() has been called")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass