

Report Title

REPORT SUBTITLE

Name | Course Title | Date

Evidence of Learning

1. Import the CSV API to read the file from Dataset

import csv

1. Declare class and its constructor

class pipelineIncident :

def \_\_init\_\_(self, Incident\_Number, Incident\_Types, Reported\_Date, Nearest\_Populated\_Centre, Province, Company, Substance, Significant, What\_happened\_category):

self.Incident\_Number = Incident\_Number

self.Incident\_Types = Incident\_Types

self.Reported\_Date = Reported\_Date

self.Nearest\_Populated\_Centre = Nearest\_Populated\_Centre

self.Province = Province

self.Company = Company

self.Substance = Substance

self.Significant = Significant

self.What\_happened\_category = What\_happened\_category

1. Declare class setters and getters

def get\_Incident\_Number(self):

"""Return Incident Number."""

return self.Incident\_Number

def set\_Incident\_Number(self, a):

"""Set Incident Number."""

self.Incident\_Number = a

def get\_Incident\_Types(self):

"""Set Incident Types."""

return self.Incident\_Types

def set\_Incident\_Types(self, a):

"""Set Incident Types."""

self.Incident\_Types = a

def get\_Reported\_Date(self):

"""Get Reported Date."""

return self.Reported\_Date

def set\_Reported\_Date(self, a):

"""Set Reported Date."""

self.Reported\_Date = a

def get\_Nearest\_Populated\_Centre(self):

"""get \_Nearest\_Populated\_Centre."""

return self.Nearest\_Populated\_Centre

def set\_Nearest\_Populated\_Centre(self, a):

"""set \_Nearest\_Populated\_Centre."""

self.Incident\_Number = a

def get\_Province(self):

"""to get\_Province."""

return self.Province

def set\_Province(self, a):

"""to set\_Province."""

self.Province = a

def get\_Company(self):

"""to get\_Company."""

return self.Company

def set\_Company(self, a):

"""to set\_Company."""

self.Company = a

def get\_Substance(self):

"""to get\_Substance ."""

return self.Substance

def set\_Substance(self, a):

"""to set\_Substance."""

self.Substance = a

def get\_Significant(self):

"""to get\_Significant ."""

return self.Significant

def set\_Significant(self, a):

"""to set\_Significant ."""

self.Significant = a

def get\_What\_happened\_category(self):

"""to get\_What\_happened\_category ."""

return self.Significant

def set\_What\_happened\_category(self, a):

"""to set\_What\_happened\_category ."""

self.What\_happened\_category = a

1. Declare a list and initialize empty

pipelineIncidentList = []

1. Read file properly and handle the exception

try:

# handle exception

with open('pipeline-incidents-comprehensive-data.csv', 'rt') as f: #read file from dataset

csv\_reader = csv.reader(f)

next(csv\_reader) # skip the heading

for line in csv\_reader: # save data into List

pipelineIncidentList.append( pipelineIncident(line[0], line[1], line[2], line[3], line[4], line[5], line[10], line[12], line[17]) )

except FileNotFoundError:

print('File not found!!', file\_name)

# If any other error occurs, print the file name

except Exception:

print('Another Error!!', file\_name)

1. Calculate the records using for loop

# calculate count of the records using for loops

totalRecords = 0

for incident in pipelineIncidentList:

totalRecords = totalRecords + 1

print(incident.get\_Incident\_Number() + " " + incident.get\_Incident\_Types() + " " + incident.get\_Reported\_Date() + " " +incident.get\_Nearest\_Populated\_Centre()+ " "+ incident.get\_Province()+ " "+incident.get\_Company()+ " "+ incident.get\_Substance()+ " "+incident.get\_Significant()+ " "+ incident.get\_What\_happened\_category())

print ('\n\n\n Total Records in this dataset is ' + str(totalRecords)+ '\n')

1. Count the incident records according to province using for loop

dictionaryProvince = {}

for incident in pipelineIncidentList: # for loop to calculate the count of incidents against each province

province = incident.get\_Province()

if province not in dictionaryProvince:

dictionaryProvince[province] = 0

dictionaryProvince[province] += 1

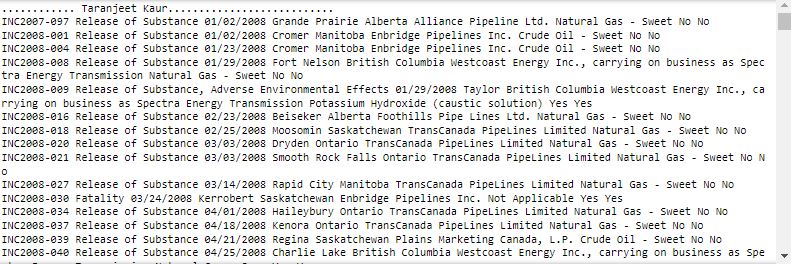
for key, value in dictionaryProvince.items(): # print the province and its count

print(key, ' : ', value)

Program Demonstration via Screen Shots

Display data using for loop

Display count of incident in each province



Source Code Commenting Example

import csv

class pipelineIncident :

"""Declaration of pipelineIncident class."""

def \_\_init\_\_(self, Incident\_Number, Incident\_Types, Reported\_Date, Nearest\_Populated\_Centre, Province, Company, Substance, Significant, What\_happened\_category):

self.Incident\_Number = Incident\_Number

self.Incident\_Types = Incident\_Types

self.Reported\_Date = Reported\_Date

self.Nearest\_Populated\_Centre = Nearest\_Populated\_Centre

self.Province = Province

self.Company = Company

self.Substance = Substance

self.Significant = Significant

self.What\_happened\_category = What\_happened\_category

def get\_Incident\_Number(self):

"""Return Incident Number."""

return self.Incident\_Number

def set\_Incident\_Number(self, a):

"""Set Incident Number."""

self.Incident\_Number = a

def get\_Incident\_Types(self):

"""Set Incident Types."""

return self.Incident\_Types

def set\_Incident\_Types(self, a):

"""Set Incident Types."""

self.Incident\_Types = a

def get\_Reported\_Date(self):

"""Get Reported Date."""

return self.Reported\_Date

def set\_Reported\_Date(self, a):

"""Set Reported Date."""

self.Reported\_Date = a

def get\_Nearest\_Populated\_Centre(self):

"""get \_Nearest\_Populated\_Centre."""

return self.Nearest\_Populated\_Centre

def set\_Nearest\_Populated\_Centre(self, a):

"""set \_Nearest\_Populated\_Centre."""

self.Incident\_Number = a

def get\_Province(self):

"""to get\_Province."""

return self.Province

def set\_Province(self, a):

"""to set\_Province."""

self.Province = a

def get\_Company(self):

"""to get\_Company."""

return self.Company

def set\_Company(self, a):

"""to set\_Company."""

self.Company = a

def get\_Substance(self):

"""to get\_Substance ."""

return self.Substance

def set\_Substance(self, a):

"""to set\_Substance."""

self.Substance = a

def get\_Significant(self):

"""to get\_Significant ."""

return self.Significant

def set\_Significant(self, a):

"""to set\_Significant ."""

self.Significant = a

def get\_What\_happened\_category(self):

"""to get\_What\_happened\_category ."""

return self.Significant

def set\_What\_happened\_category(self, a):

"""to set\_What\_happened\_category ."""

self.What\_happened\_category = a

pipelineIncidentList = []

try: # handle exception

with open('pipeline-incidents-comprehensive-data.csv', 'rt') as f: #read file from dataset

csv\_reader = csv.reader(f)

next(csv\_reader) # skip the heading

for line in csv\_reader: # save data into List

pipelineIncidentList.append( pipelineIncident(line[0], line[1], line[2], line[3], line[4], line[5], line[10], line[12], line[17]) )

except FileNotFoundError:

print('File not found!!', file\_name)

# If any other error occurs, print the file name

except Exception:

print('Another Error!!', file\_name)

print('............ Taranjeet Kaur...........................')

# calculate count of the records using for loops

totalRecords = 0

for incident in pipelineIncidentList:

totalRecords = totalRecords + 1

print(incident.get\_Incident\_Number() + " " + incident.get\_Incident\_Types() + " " + incident.get\_Reported\_Date() + " " +incident.get\_Nearest\_Populated\_Centre()+ " "+ incident.get\_Province()+ " "+incident.get\_Company()+ " "+ incident.get\_Substance()+ " "+incident.get\_Significant()+ " "+ incident.get\_What\_happened\_category())

print ('\n\n\n Total Records in this dataset is ' + str(totalRecords)+ '\n')

print('............ Taranjeet Kaur...........................')

#Count of Incident according to province

dictionaryProvince = {}

for incident in pipelineIncidentList: # for loop to calculate the count of incidents against each province

province = incident.get\_Province()

if province not in dictionaryProvince:

dictionaryProvince[province] = 0

dictionaryProvince[province] += 1

for key, value in dictionaryProvince.items(): # print the province and its count

print(key, ' : ', value)