Assignment 1: -  
# # 👉🏼👉🏼 Task 1: Declare two numeric variables and perform arithmetic operations like addition,  
# number\_1 = 45  
# number\_2 = 63  
#  
# # performing arithmetic operations like addition, subtraction, multiplication and division  
# addition = (number\_1 + number\_2)  
# subtraction = (number\_1 - number\_2)  
# multiplication = (number\_1 \* number\_2)  
# division = (number\_1/number\_2)  
#  
# print('addition = {}'.format(addition))  
# print('subtraction = {}'.format(subtraction))  
# print('multiplication = {}'.format(multiplication))  
# print('division = {}'.format(division))  
# print("Process Completed")  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 👉🏼👉🏼 Task 2: Loop through a list of five integers and print the integer only if it is greater than 50.  
# Create a list  
# number\_list = [98, 49, 58, 100, 10, 5, 60, 53, 70, 35, 25, 45]  
# print("printing all integer ")  
# for l in number\_list:  
# print(l)  
#  
# print("printing the integer only if it is greater than 50")  
# for y in number\_list:  
# if y > 50:  
# print(y)  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
*"""  
👉🏼👉🏼 Task 3: Creating Buffers:  
a. Create a 500-meter buffer using the Python window  
b. Create 1000, 1200, and 1400 feet buffers in a single Python script using an IDE and verify the results in ArcGIS Pro  
"""*# Create a 500-meter buffer using the Python window \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# Import arcpy package  
import arcpy  
  
  
# Setting the default workspace  
# arcpy.env.workspace = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_1\ProProject\_Practical\_One\Practical\_One.gdb"  
  
# Corrected usage of arcpy.analysis.Buffer  
# arcpy.analysis.Buffer("Wilson\_Schools", "Wilson\_School\_Buffer\_500m", "500 Meters")  
# print("Process Completed")  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# Create 1000, 1200, and 1400 feet buffers in a single Python script using an IDE and verify the results in ArcGIS Pro  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# # Setting the default workspace  
# arcpy.env.workspace = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_1\ProProject\_Practical\_One\Practical\_One.gdb"  
#  
# # Create 1000 feet buffer  
# arcpy.analysis.Buffer("Wilson\_Schools", "Wilson\_School\_Buffer\_1000ft", "1000 Feet")  
#  
# # Create 1200 feet buffer  
# arcpy.analysis.Buffer("Wilson\_Schools", "Wilson\_School\_Buffer\_1200ft", "1200 Feet")  
#  
# # Create 1400 feet buffer  
# arcpy.analysis.Buffer("Wilson\_Schools", "Wilson\_School\_Buffer\_1400ft", "1400 Feet")  
#  
# print("Process Completed")  
  
# ======================================================================================================================  
'''  
👉🏼👉🏼 Task 4: Use Arcpy. management.FeatureToPoint to convert Wilson\_Zoning   
 polygon feature class into a point feature class using an IDE and verify the results in ArcGIS Pro  
'''  
# Import arcpy package  
import arcpy  
  
# Setting the default workspace  
Wilson\_Zoning\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_1\ProProject\_Practical\_One\Practical\_One.gdb\Wilson\_Zoning"  
  
# Create point features at the centroids of polygons  
wilson\_polygon = Wilson\_Zoning\_path + "\_PolygonToPoint"  
arcpy.management.FeatureToPoint(Wilson\_Zoning\_path, wilson\_polygon )  
  
print("Process Completed")

Assignment 2: -

import arcpy  
import os  
  
# Question 1: What is the syntax for the describe tool? \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# # Specify the path to the feature class  
# fc\_path = r'D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_2\Data\ProProject\_Practical\_Two\World\_data.gdb\Lakes'  
#  
# # Describe the feature class  
# desc = arcpy.Describe(fc\_path)  
#  
# # Access and print specific properties  
# print(f"Feature Class Name: {desc.name}")  
# print(f"Data Type: {desc.dataType}")  
# print(f"Shape Type: {desc.shapeType}")  
  
# Quistion 2: How can you obtain spatial reference name for a feature class? \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# # Specify the path to the feature class  
# feature\_path = r'D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_2\Data\ProProject\_Practical\_Two\World\_data.gdb\Lakes'  
#  
# # Describe the feature class  
# desc = arcpy.Describe(feature\_path)  
#  
# # Access the spatial reference name  
# spatial\_reference\_name = desc.spatialReference.name  
#  
# # Print the spatial reference name  
# print("Spatial Reference Name: " + spatial\_reference\_name)  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Task 1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# Q1. Create a list object which contains name of different cities and perform the following operations on it:  
# Q1.1. Append a new city to the existing list  
# ----------------------------------------------------------------------------------------------------------------  
# # Create a list of city names  
# cities = ["Kolkata", "Delhi", "Mumbai", "Chennai", "Darjeeling"]  
#  
# # Append a new city to the list  
# new\_city = "Pune"  
# cities.append(new\_city)  
#  
# # Print the updated list  
# print(cities)  
  
# -----------------------------------------------------------------------------------------------------------------  
# Q1.2. Print the length of the list  
# -----------------------------------------------------------------------------------------------------------------  
# # Create a list of city names  
# cities = ["Kolkata", "Delhi", "Mumbai", "Chennai", "Darjeeling"]  
#  
# # Append a new city to the list  
# new\_city = "Pune"  
# cities.append(new\_city)  
#  
# # # Print the updated list  
# print(cities)  
#  
# # Print the length of the list  
# print("Length of the list:", len(cities))  
  
# -----------------------------------------------------------------------------------------------------------------  
# Q1.3. Remove an element at index '2’ from the list and print  
# -----------------------------------------------------------------------------------------------------------------  
  
# # Create a list of city names  
# cities = ["Kolkata", "Delhi", "Mumbai", "Chennai", "Darjeeling"]  
#  
# # Remove the element at index 2 (third element)  
# removed\_city = cities.pop(2)  
#  
# # Print the updated list  
# print("Updated list after removing element at index 2:")  
# print(cities)  
#  
# # Print the removed city  
# print("Removed city:", removed\_city)  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# Q2. Create a dictionary object which contains country name as key and their capital as value and perform the following operations on it:  
# Q2.1 Print only the name of countries (keys)  
# ----------------------------------------------------------------------------------------------------------------------------------------  
# # Create a dictionary of countries and capitals  
# country\_capital\_dict = {  
# "India": "Delhi",  
# "Canada": "Ottawa",  
# "United Kingdom": "London",  
# "France": "Paris",  
# "Germany": "Berlin"  
# }  
#  
# # Print only the names of countries (keys)  
# country\_names = list(country\_capital\_dict.keys())  
# print("Names of Countries:")  
# for country in country\_names:  
# print(country)  
  
# ----------------------------------------------------------------------------------------------------------------------  
# Q2.2. Insert one more country and its capital  
# ----------------------------------------------------------------------------------------------------------------------  
  
# Existing dictionary of countries and capitals  
# country\_capital\_dict = {  
# "India": "Delhi",  
# "Canada": "Ottawa",  
# "United Kingdom": "London",  
# "France": "Paris",  
# "Germany": "Berlin"  
# }  
#  
# # Insert one more country and its capital  
# new\_country = "Italy"  
# new\_capital = "Rome"  
# country\_capital\_dict[new\_country] = new\_capital  
#  
# # Print the updated dictionary  
# country\_names = list(country\_capital\_dict.keys())  
# print("Names of Countries:")  
# for country in country\_names:  
# print(country)  
  
# -----------------------------------------------------------------------------------------------------------------------  
# Print the length of the dictionary  
# -----------------------------------------------------------------------------------------------------------------------  
# # Create a dictionary of countries and capitals  
# country\_capital\_dict = {  
# "India": "Delhi",  
# "Canada": "Ottawa",  
# "United Kingdom": "London",  
# "France": "Paris",  
# "Germany": "Berlin"  
# }  
# # Print the length of the dictionary  
# dictionary\_length = len(country\_capital\_dict)  
# print("Length of the Dictionary:", dictionary\_length)  
  
# -----------------------------------------------------------------------------------------------------------------------  
# Remove an element from the dictionary  
# -----------------------------------------------------------------------------------------------------------------------  
# # Existing dictionary of countries and capitals  
# country\_capital\_dict = {  
# "India": "Delhi",  
# "Canada": "Ottawa",  
# "United Kingdom": "London",  
# "France": "Paris",  
# "Germany": "Berlin"  
# }  
#  
# # Remove a specific element by key  
# remove = "Canada"  
# if remove in country\_capital\_dict:  
# del country\_capital\_dict[remove]  
#  
#  
# # Print the updated dictionary  
# country\_names = list(country\_capital\_dict.keys())  
# print("Names of Countries:")  
# for country in country\_names:  
# print(country)  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# Task 2: Describe the “Lakes” feature class from the world\_data.gdb and print:  
# a. Spatial reference  
# ----------------------------------------------------------------------------------------------------------------------------  
  
# # Specify the path to the feature class  
# fc\_path = r'D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_2\Data\ProProject\_Practical\_Two\World\_data.gdb\Lakes'  
#  
# # Describe the feature class  
# desc = arcpy.Describe(fc\_path)  
#  
# # Access and print specific properties  
# print(f"Spatial Reference Name: {desc.spatialReference.name}")  
  
# -----------------------------------------------------------------------------------------------------------------------  
# Feature type  
# -----------------------------------------------------------------------------------------------------------------------  
# # Specify the path to the feature class  
# fc\_path = r'D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_2\Data\ProProject\_Practical\_Two\World\_data.gdb\Lakes'  
#  
# # Describe the feature class  
# desc = arcpy.Describe(fc\_path)  
#  
# # Access and print specific properties  
# print(f"Shape Type: {desc.shapeType}")  
  
# -----------------------------------------------------------------------------------------------------------------------  
# Field names and its respective type.  
# -----------------------------------------------------------------------------------------------------------------------  
  
# # Specify the path to the feature class  
# fc\_path = r'D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_2\Data\ProProject\_Practical\_Two\World\_data.gdb\Lakes'  
#  
# # Describe the feature class  
# desc = arcpy.Describe(fc\_path)  
#  
# # Access and print specific properties  
# print(f"Feature Class Name: {desc.name}")  
# print(f"Data Type: {desc.dataType}")  
  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# Task 3: Describe the “erelev” raster from the RASTER\_DATA folder and print the following:  
# Band count  
# -----------------------------------------------------------------------------------------------------------------------------  
# # Specify the path to the raster dataset  
# raster\_path = r'D:\1\_BVIEER\3rd\_sem\Programming For GIS II\Practical\_1\Ladsat\_data\_from\_GEE\Purbasthali\_LS7\_data.tif'  
#  
# # Describe the raster dataset  
# desc = arcpy.Describe(raster\_path)  
#  
# # Print the requested information  
# print(f"Band Count: {desc.bandCount}")  
  
# --------------------------------------------------------------------------------------------------------------------------  
# Format  
# --------------------------------------------------------------------------------------------------------------------------  
  
# # Specify the path to the raster dataset  
# raster\_path = r'D:\1\_BVIEER\3rd\_sem\Programming For GIS II\Practical\_1\Ladsat\_data\_from\_GEE\Purbasthali\_LS7\_data.tif'  
#  
# # Describe the raster dataset  
# desc = arcpy.Describe(raster\_path)  
#  
# # Print the requested information  
# print(f"Format: {desc.format}")  
  
# -----------------------------------------------------------------------------------------------------------------------  
# Image Base Name  
# -----------------------------------------------------------------------------------------------------------------------  
# # Specify the path to the raster dataset  
# raster\_path = r"D:\1\_BVIEER\3rd\_sem\Programming For GIS II\Practical\_1\Ladsat\_data\_from\_GEE\Purbasthali\_LS7\_data.tif"  
#  
# # Describe the raster dataset  
# desc = arcpy.Describe(raster\_path)  
#  
# # Print the requested information  
# print(f"Base Name: {desc.basename}")  
  
# ------------------------------------------------------------------------------------------------------------------------  
# Image height and width  
# ------------------------------------------------------------------------------------------------------------------------  
# Create a Describe object from the raster band  
desc = arcpy.Describe(r"D:\1\_BVIEER\3rd\_sem\Programming For GIS II\Practical\_2\Output\KALNA\_2011\_NDVI.tif")  
  
# Print some raster band properties  
  
print("Height: %d" % desc.height)  
print("Width: %d" % desc.width)  
  
  
  
  
  
  
Assignment 3: -

import arcpy  
import os  
  
# # Set the workspace (replace with your workspace path)  
# arcpy.env.workspace= 'D:/1\_BVIEER/3rd\_sem/Programming for GIS- III Mr. Ronit Jadhav/Assignments/Assignment\_3/ProProject\_AutomatingUsingLists/ProProject\_AutomatingUsingLists.gdb'  
#  
# # Get a list of feature classes in the workspace  
# feature\_classes = arcpy.ListFeatureClasses()  
#  
# # Iterate through the list and print the names  
# for feature\_class in feature\_classes:  
# print(feature\_class)  
  
  
# --------------------------------------------------------------------------------------------------------------------------------------------------  
# Task 1: Open an already existing text file and print every line.  
# --------------------------------------------------------------------------------------------------------------------------------------------------  
# # Specify the path to the text file  
# file\_path = 'D:/1\_BVIEER/3rd\_sem/Programming for GIS- III Mr. Ronit Jadhav/Assignments/Assignment\_3/tobu.txt'  
#  
# # Open the file in read mode with 'utf-8' encoding  
# try:  
# with open(file\_path, 'r', encoding='utf-8') as file:  
# for line in file:  
# print(line, end="")  
# except FileNotFoundError:  
# print(f"The file '{file\_path}' does not exist")  
# except Exception as e:  
# print(f"An error occurred: {e}")  
  
  
# --------------------------------------------------------------------------------------------------------------------------------------------------  
# Task 2: Document field properties for the “MajorAttractions” feature class into a text file by following the below workflow:  
# --------------------------------------------------------------------------------------------------------------------------------------------------  
# # Step 1: Import required modules and set workspace  
# arcpy.env.workspace = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_3\ProProject\_AutomatingUsingLists\ProProject\_AutomatingUsingLists.gdb"  
#  
# # Step 2: Create a list of fields from the feature class  
# feature\_class\_name = "MajorAttractions"  
# fields = arcpy.ListFields(feature\_class\_name)  
#  
# # Step 3: Open a new text file and add header lines  
# output\_folder = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_3"  
# output\_file\_name = "MajorAttractions\_FieldInfo.txt"  
# output\_file\_path = os.path.join(output\_folder, output\_file\_name)  
#  
# with open(output\_file\_path, 'w') as file\_obj:  
# file\_obj.write(f"Field properties for the '{feature\_class\_name}' feature class:\n\n")  
#  
# # Step 4: Loop through the list of fields  
# for field in fields:  
# # Step 5: Write each field's properties to the text file  
# file\_obj.write(f"Field Name: {field.name}\n")  
# file\_obj.write(f"Data Type: {field.type}\n")  
# file\_obj.write(f"Length: {field.length}\n")  
# file\_obj.write(f"Scale: {field.scale}\n")  
# file\_obj.write("\n")  
# # Step 6: Indicate that the script is completed  
# print(f"Field properties have been written to {output\_file\_path}")  
  
  
# --------------------------------------------------------------------------------------------------------------------------------------------------  
'''  
 Write a script that buffers feature classes, with the buffer distance determined by the type of geometry. For point feature, the buffer distance   
 will be set to one value; for polyline features, the buffer distance will be set to a second value; for polygon features, the buffer distance   
 will be set to a third value.  
'''  
# --------------------------------------------------------------------------------------------------------------------------------------------------  
# Set the workspace  
arcpy.env.workspace = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_3\ProProject\_AutomatingUsingLists\ProProject\_AutomatingUsingLists.gdb"  
# List all feature classes in the workspace  
fc\_list = arcpy.ListFeatureClasses()  
# Loop through each feature class  
for fc in fc\_list:  
 # Describe the feature class to get its shape type  
 desc\_obj = arcpy.Describe(fc)  
 shape\_type = desc\_obj.shapeType  
  
 # Set buffer distances based on shape type  
 if shape\_type == "Point":  
 buffer\_distance = 450 # 450 feet  
 elif shape\_type == "Polyline":  
 buffer\_distance = 1500 # 1500 feet  
 elif shape\_type == "Polygon":  
 buffer\_distance = 500 # 500 feet  
 # Create an output buffer feature class  
 output\_buffer = fc + "\_Buffer"  
 # Perform the buffer analysis using the specified buffer distance  
 arcpy.analysis.Buffer(fc, output\_buffer, f"{buffer\_distance} feet")  
 print(f"Buffer created for {fc} with a {buffer\_distance} buffer distance.")  
 print("Process Completed")

Assignment 4: -

import arcpy  
import os  
  
  
# Question 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# arcpy.management.SelectLayerByAttribute("Population", "NEW\_SELECTION", "AGE = 1")  
# arcpy.management.SelectLayerByLocation("State", "WITHIN\_A\_DISTANCE", "restaurant\_lyr", "1000 feet")  
  
# Question 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# output\_restr\_within\_dist = "Wilson\_within\_histdist\_restaurant"  
# output\_restr\_within\_dist\_path = os.path.join(gdb\_path, output\_restr\_within\_dist)  
# arcpy.management.CopyFeatures("restaurant\_lyr", output\_restr\_within\_dist\_path)  
  
# Task 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
gdb\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Practical\_4\Given\_data\ProProject\_Selections\ProProject\_Selections.gdb"  
restaurant\_fc\_name = "Wilson\_Restaurants"  
  
restaurant\_fc\_path = os.path.join(gdb\_path, restaurant\_fc\_name)  
  
# Feature class to feature layer  
arcpy.management.MakeFeatureLayer(restaurant\_fc\_path,"restaurant\_Layer")  
  
# Count of all the features before selection  
pre\_count = arcpy.GetCount\_management("restaurant\_Layer")  
  
print("Total number of restaurants = {} ".format(pre\_count))  
  
# Task 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
gdb\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Practical\_4\Given\_data\ProProject\_Selections\ProProject\_Selections.gdb"  
restaurant\_fc\_name = "Wilson\_Restaurants"  
crime\_fc\_name = "Wilson\_Crimes96"  
  
restaurant\_fc\_path = os.path.join(gdb\_path,restaurant\_fc\_name)  
crime\_fc\_path = os.path.join(gdb\_path,crime\_fc\_name)  
  
arcpy.management.MakeFeatureLayer(restaurant\_fc\_path,"restaurant\_lyr")  
arcpy.management.MakeFeatureLayer(crime\_fc\_path,"crime\_lyr")  
  
# getting count of all the features before selection  
pre\_count1=arcpy.GetCount\_management("restaurant\_lyr")  
pre\_count3=arcpy.GetCount\_management("crime\_lyr")  
  
# Query  
arcpy.management.SelectLayerByLocation("crime\_lyr","WITHIN\_A\_DISTANCE","restaurant\_lyr","500 meters")  
post\_count3 =arcpy.GetCount\_management("crime\_lyr")  
print('crime near 500 meters of restaurant = {}'.format(post\_count3))  
  
print("Process Completed")  
  
# Task 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
gdb\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Practical\_4\Given\_data\ProProject\_Selections\ProProject\_Selections.gdb"  
restaurant\_fc\_name = "Wilson\_Restaurants"  
histdist\_fc\_name = "Wilson\_Histdist"  
crime\_fc\_name = "Wilson\_Crimes96"  
  
restaurant\_fc\_path = os.path.join(gdb\_path,restaurant\_fc\_name)  
histdist\_fc\_path = os.path.join(gdb\_path,histdist\_fc\_name)  
crime\_fc\_path = os.path.join(gdb\_path,crime\_fc\_name)  
   
arcpy.management.MakeFeatureLayer(restaurant\_fc\_path,"restaurant\_lyr")  
arcpy.management.MakeFeatureLayer(histdist\_fc\_path,"histdist\_lyr")  
arcpy.management.MakeFeatureLayer(crime\_fc\_path,"crime\_lyr")  
  
# getting count of all the features before selection  
pre\_count1=arcpy.GetCount\_management("restaurant\_lyr")  
pre\_count2=arcpy.GetCount\_management("histdist\_lyr")  
pre\_count3=arcpy.GetCount\_management("crime\_lyr")  
  
print('Total Restaurants before selection = {}'.format(pre\_count1[0]))  
print('Total historic district before selection = {}'.format(pre\_count2[0]))  
print('Total crime before selection = {}'.format(pre\_count3[0]))  
  
arcpy.management.SelectLayerByAttribute("restaurant\_lyr","NEW\_SELECTION","ALCOHOL=1")  
post\_count1 = arcpy.GetCount\_management("restaurant\_lyr")  
print('total restaurants serves alcohol = {}'.format(post\_count1))  
  
arcpy.management.SelectLayerByLocation("restaurant\_lyr","WITHIN\_A\_DISTANCE","histdist\_lyr","1000 feet","SUBSET\_SELECTION")  
post\_count2 =arcpy.GetCount\_management("restaurant\_lyr")  
print('total restaurants serves alcohol within 1000 feet of histdist = {}'.format(post\_count2))  
  
arcpy.management.SelectLayerByLocation("crime\_lyr","WITHIN\_A\_DISTANCE","restaurant\_lyr","500 feet")  
post\_count3 =arcpy.GetCount\_management("crime\_lyr")  
print('crime near 500 feet of restaurant = {}'.format(post\_count3))  
  
arcpy.management.SelectLayerByAttribute("crime\_lyr","SUBSET\_SELECTION","ALCOHOL>0")  
post\_count4 = arcpy.GetCount\_management("crime\_lyr")  
print('Alcohol related crime = {}'.format(post\_count4))  
  
  
print("Process Completed")

Assignment 5: -

# Task 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
import arcpy  
import os  
gdp\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_5\ProProject\_Cursors\ProProject\_Cursors\ProProject\_Cursors.gdb"  
fc\_name = "MajorAttractions"  
fc\_path = os.path.join(gdp\_path,fc\_name)  
  
field\_list = ["NAME","ADDR"]  
  
with arcpy.da.SearchCursor(fc\_path,field\_list) as s\_cursor:  
 for row in s\_cursor:  
 print("Name:{} | Address:{}".format(row[0], row[1]))  
 del s\_cursor  
  
# Task 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# import arcpy  
# import os  
gdp\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_5\ProProject\_Cursors\ProProject\_Cursors\ProProject\_Cursors.gdb"  
  
fc\_name = "MajorAttractions"  
fc\_path = os.path.join(gdp\_path,fc\_name)  
  
field\_list = ["NAME","ADDR","ESTAB"]  
  
with arcpy.da.SearchCursor(fc\_path,field\_list,"ESTAB>1970") as s\_cursor:  
 for row in s\_cursor:  
 print("Name:{}, Establishment:{}".format(row[0], row[2]))  
 del s\_cursor  
print("Process completed")  
  
# Task 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
import arcpy  
import os  
gdp\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_5\ProProject\_Cursors\ProProject\_Cursors\ProProject\_Cursors.gdb"  
  
fc\_name = "MajorAttractions"  
fc\_path = os.path.join(gdp\_path,fc\_name)  
  
field\_list = ["NAME","SHAPE@X","SHAPE@Y"]  
print("-------------------------- majorAttraction and its coordinates -------------------------------")  
with arcpy.da.SearchCursor(fc\_path,field\_list) as s\_cursor:  
 for row in s\_cursor:  
 print("Name: {}, X-corrdinates:{}, Y-coordinates:{}".format(row[0],row[1], row[2]))  
 del s\_cursor

Assignment 6: -

# Task 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
import arcpy  
import os  
gdp\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_5\ProProject\_Cursors\ProProject\_Cursors\ProProject\_Cursors.gdb"  
fc\_name = "MajorAttractions"  
  
fc\_path = os.path.join(gdp\_path,fc\_name)  
  
field\_list = ["NAME","ESTAB","HISTORIC"]  
  
with arcpy.da.UpdateCursor(fc\_path,field\_list) as U\_cursor:  
 for row in U\_cursor:  
  
 if row[1]<1960:  
 row[2] = "yes"  
 else:  
 row[2] = "No"  
 U\_cursor.updateRow(row)  
  
print("Process Completed")  
  
# Task 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
gdp\_path = r"D:\1\_BVIEER\3rd\_sem\Programming for GIS- III Mr. Ronit Jadhav\Assignments\Assignment\_5\ProProject\_Cursors\ProProject\_Cursors\ProProject\_Cursors.gdb"  
fc\_name = "MajorAttractions"  
  
fc\_path = os.path.join(gdp\_path,fc\_name)  
  
field\_list = ["NAME","ESTAB","ADDR","CITYNM","ZIP","EMP","ACRES"]  
  
record\_1 = (" Five Star Restaurant",2021,"841 STREET","SAN DIEGO",92101,150,10)  
record\_2 = (" New Town Restaurant ",2022,"842 STREET","SAN DIEGO",92105,140,15)  
  
i\_cursor = arcpy.da.InsertCursor(fc\_path,field\_list)  
i\_cursor.insertRow(record\_1)  
i\_cursor.insertRow(record\_2)  
print("Process Completed")