**CSCI 333 Final Project**

**Task1:**

**Code:**

**class Pet:**

**def \_\_init\_\_(self):**

**self.name = ""**

**self.animal\_type = ""**

**self.age = 0**

**def set\_name(self, name):**

**self.name = name**

**def set\_animal\_type(self, animal\_type):**

**self.animal\_type = animal\_type**

**def set\_age(self, age):**

**self.age = age**

**def get\_name(self):**

**return self.name**

**def get\_animal\_type(self):**

**return self.animal\_type**

**def get\_age(self):**

**return self.age**

**def main():**

**# Create a Pet object**

**pet = Pet()**

**# Prompt the user to enter pet details**

**name = input("Enter pet's name: ")**

**animal\_type = input("Enter pet's animal type: ")**

**age = int(input("Enter pet's age: "))**

**# Set pet's attributes using the setter methods**

**pet.set\_name(name)**

**pet.set\_animal\_type(animal\_type)**

**pet.set\_age(age)**

**# Display pet's details using accessor methods**

**print("\nHere are your Pet's Details:")**

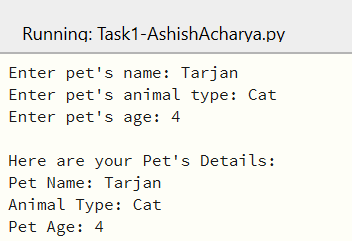
**print(" Pet Name:", pet.get\_name())**

**print("Animal Type:", pet.get\_animal\_type())**

**print("Pet Age:", pet.get\_age())**

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

**main()**

**Output:**  


**Task2:**

**Code:**

**import pandas as pd**

**# a. Creating a DataFrame named temperatures from a dictionary of three temperature readings each for three people ’Maxine’, ’James’ and ’Amanda’.**

**data = {'Maxine': [25, 28, 24],**

**'James': [22, 24, 20],**

**'Amanda': [27, 26, 23]}**

**temperatures = pd.DataFrame(data)**

**# b. Recreate the DataFrame temperatures in Part (a) with custom indices using the index keyword argument and a list containing ’Morning’, ’Afternoon’ and ’Evening’.**

**custom\_indices = ['Morning', 'Afternoon', 'Evening']**

**temperatures\_custom\_index = pd.DataFrame(data, index=custom\_indices)**

**# c. Select from temperatures the column of temperature readings for ’Maxine’.**

**maxine\_temperatures = temperatures['Maxine']**

**# d. Select from temperatures the row of ’Morning’ temperature readings.**

**morning\_temperatures = temperatures\_custom\_index.loc['Morning']**

**# e. Select from temperatures the rows for ’Morning’ and ’Evening’ temperature readings.**

**morning\_evening\_temperatures = temperatures\_custom\_index.loc[['Morning', 'Evening']]**

**# f. Select from temperatures the columns of temperature readings for ’Amanda’ and ’Maxine’.**

**amanda\_maxine\_temperatures = temperatures\_custom\_index[['Amanda', 'Maxine']]**

**# g. Select from temperatures the elements for ’Amanda’ and ’Maxine’ in the ’Morning’ and ’Afternoon’**

**amanda\_maxine\_morning\_afternoon = temperatures\_custom\_index.loc[['Morning', 'Afternoon'], ['Amanda', 'Maxine']]**

**# h. Use the describe() method to produce temperatures’ descriptive statistics**

**temperatures\_describe = temperatures\_custom\_index.describe()**

**# i. Transpose the temperatures**

**temperatures\_transposed = temperatures\_custom\_index.transpose()**

**# j. Sort temperatures so that its column names are in alphabetical order.**

**temperatures\_sorted = temperatures\_custom\_index.sort\_index(axis=1)**

**# Print the results**

**print("(a) DataFrame temperatures:")**

**print()**

**print(temperatures)**

**print("\n(b) DataFrame temperatures with custom indices:")**

**print(temperatures\_custom\_index)**

**print("\n(c) Maxine's temperatures:")**

**print(maxine\_temperatures)**

**print("\n(d) Morning temperatures:")**

**print(morning\_temperatures)**

**print("\n(e) Morning and Evening temperatures:")**

**print(morning\_evening\_temperatures)**

**print("\n(f) Amanda and Maxine's temperatures:")**

**print(amanda\_maxine\_temperatures)**

**print("\n(g) Amanda and Maxine's Morning & Afternoon temperatures:")**

**print(amanda\_maxine\_morning\_afternoon)**

**print("\n(h) Descriptive statistics for temperatures:")**

**print(temperatures\_describe)**

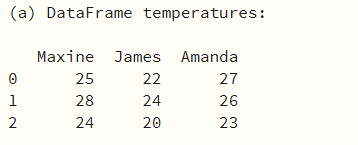
**print("\n(i) Transposed temperatures:")**

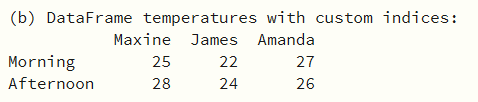
**print(temperatures\_transposed)**

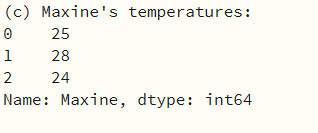
**print("\n(j) Sorted temperatures:")**

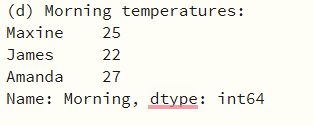
**print(temperatures\_sorted)**

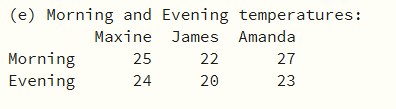
**Output:**

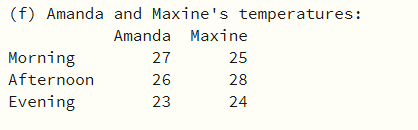


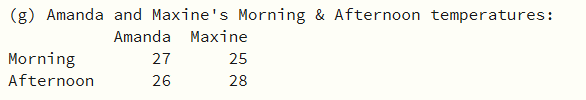


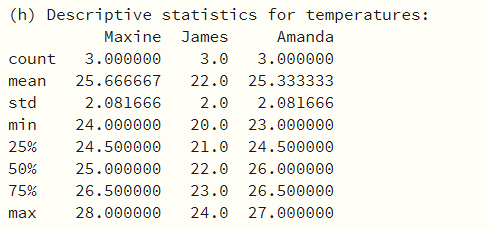


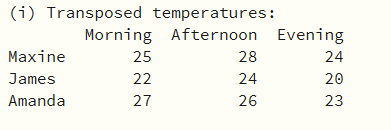


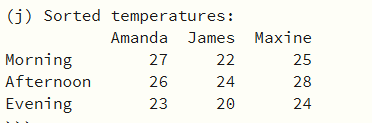












**Task3:**

**Code:**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import load\_digits**

**from sklearn.model\_selection import train\_test\_split**

**# Load the Digits dataset**

**digits = load\_digits()**

**# Task (a): Display the two-dimensional array and numeric value of the digit at index 35**

**sample\_image = digits.images[35]**

**target\_value = digits.target[35]**

**print("Sample Image:")**

**print(sample\_image)**

**print("Target Value:", target\_value)**

**# Task (b): Display the image at index 35 of the Digits dataset**

**plt.imshow(sample\_image, cmap='gray')**

**plt.title(f"Sample Image at Index 35 (Digit {target\_value})")**

**plt.show()**

**# Task (c): Train-test split and numbers of training/testing samples**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**digits.data, digits.target, random\_state=11, test\_size=0.70**

**)**

**num\_train\_samples = len(X\_train)**

**num\_test\_samples = len(X\_test)**

**print("Number of Training Samples:", num\_train\_samples)**

**print("Number of Testing Samples:", num\_test\_samples)**

**# Task (d): Display the number of training examples and testing examples**

**print("Number of Training Examples:", num\_train\_samples)**

**print("Number of Testing Examples:", num\_test\_samples)**

**# Task (e): Rewrite the list comprehension using a for loop**

**predicted\_values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]**

**expected\_values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 1]**

**wrong\_predictions = []**

**for predicted, expected in zip(predicted\_values, expected\_values):**

**if predicted != expected:**

**wrong\_predictions.append((predicted, expected))**

**# Task (f): Explanation for row 3 of the confusion matrix**

**confusion\_row\_values = [10, 5, 120, 0, 3, 2, 0, 1, 8, 1]**

**print("Explanation for confusion matrix row 3:")**

**for i, count in enumerate(confusion\_row\_values):**

**print(f"Predicted as {i}: {count} times")**

**Output:**

