PSET7-ObjectOriented

April 16, 2025

```
<h2 style ="text-align:left; padding-top:5px;"> CS 101 - Foundation of Data Science and Enginee
<p><b> PSET-7 - Object Oriented Programming
```

0.0.1 Objective:

In this assignment you will be working with Class, Objects and Inheritance that was discussed in the class.

0.0.2 This is an individual assignment. No collaboration is allowed!

0.1 DataAnalyzer for Student Projects

As a data science student, you are tasked with analyzing project data for a research lab. The lab wants to track the progress and results of various student projects to optimize resource allocation and promote effective study habits. Your program will need to manage project data and calculate statistics based on this data.

0.1.1 Question 1: (5 pts)

Create a class called Student with the following attributes

1.student_id: a unique identifier for the student as an integer

2.major: the student's major as a string (e.g., "Computer Science", "Data Science")

3.university: the name of the university (e.g., "Harvard", "MIT", "Other")

Please be sure to include a constructor that can initialize the class with the above attributes. Assume inputs passed to your class will always be correct i.e. you are not required to check types, and verify uppercase or lower case.

[9]:
```python
class Student:
    def __init__(self, student_id, major, university):
        self.student_id = student_id
        self.major = major
        self.university = university

#arguents are student_id - a unique identifier for the student as an integer,␣
 ↪major - the student's major as a string, university - the name of the␣
 ↪university
```

0.1.2 Question 2: (50 pts)

Create a class called Project which is a subclass of Student with the following attributes and methods.

0.1.3 Attributes (5 points) :

Public attributes :

1. project_id: a unique identifier for the project as an integer

2. data_points: a list of numerical values collected from the project

Private attributes : (In python private members are stored by adding 2 underscores in front of the name of the variable. Example: analysis_results should be declared as ___analysis_results)

3. analysis_results : a dictionary to store results of various analysis

4. active : a boolean indicating if the project is currently active (default is True)

0.1.4 Methods 1-4(15 points) :

1. add_data(self, new_data): Appends a new data point to the data_points list.

2. get_results(self): Returns the analysis_results dictionary

3. is_active(self): Returns the status of the active attribute.

4. set_active(self,status): Sets the status of the active attribute (True/False).

5. perform_analysis(self): Analyzes data_points to calculate and store/update analysis_results with statistical results (mean, median, variance). (20 pts)

Notes for perform_analysis:

1. The method should only store the mean, median, and variance in analysis_results dictionary.It should not return the analysis_results.

2. Use the sample formula to calculate variance.

3. Use of inbuilt methods is not allowed for calculation of mean, median and variance

4. Please do not omit the decimal points for each of the statistic calculated.

Exception Handling :(10 points)

1. Implement a mechanism to handle incorrect data type. At the minimum you should check all the data in the list are numerical in the constructor, and check if new data point being added is of numerical type. Raise an error if you encounter list that have non-numerical values and if non-numerical values are being added to the list. https://docs.python.org/3/library/exceptions.html#TypeError (5 pts)

2. In your perform_analysis method use try-except to catch errors resulting from ZeroDivisionError if the list is empty. You should print a friendly text if you encounter such error in perform_analysis method, i.e. do not raise an error. (5 points)

Note: Please see Question 3 for order of parameters being passed to the constructor in the example

[36]:
```
class Project(Student):
    def __init__(self, student_id, major, university, project_id, data_points):
        super().__init__(student_id, major, university)
        self.project_id = project_id
        self.data_points = data_points
        self.__analysis_results = {}
        self.__active = True

    def add_data(self, new_data):
            #add numerical data to the data_points list
            #arguments: new_data (int, float, or list): A number or list of␣
 ↪numbers
            #handle incorrect data types: if the input is not numeric or a list␣
 ↪of numerics

        if isinstance(new_data, (int, float)):
            self.data_points.append(float(new_data))

        elif isinstance(new_data, list):
            if all(isinstance(item, (int, float)) for item in new_data):
                self.data_points.extend([float(x) for x in new_data])
            else:
                raise TypeError("All elements in the list must be numeric.")

        else:
            raise TypeError("Input must be a number or a list of numbers.")

    def set_active(self, status):
        #set the active status of the project
        #arguments: status (bool): True to activate, False to deactivate
        #handle incorrect data types: if status is not a boolean

        if not isinstance(status, bool):
            raise ValueError("Status must be a boolean.")
        self.__active = status

    def is_active(self):
        #check whether the project is currently active

        return self.__active

    def perform_analysis(self):
        #analyze the data_points to calculate mean, median, and variance
        #stores the result in __analysis_results
        #if data_points is empty print an error message

        try:
            n = len(self.data_points)
            if n == 0:
                raise ZeroDivisionError("No data points available for analysis.
↪")


            #mean
            total = sum(self.data_points)
            mean = total / n

            #median
            sorted_data = sorted(self.data_points)
            if n % 2 == 1:
                median = float(sorted_data[n // 2])
            else:
                mid1 = sorted_data[n // 2 - 1]
                mid2 = sorted_data[n // 2]
                median = (mid1 + mid2) / 2.0

            #sample variance
            squared_diffs = [(x - mean) ** 2 for x in self.data_points]
            if n == 1:
                variance = 0.0
            else:
                variance = sum(squared_diffs) / (n - 1)

            self.__analysis_results = {
                "mean": round(mean, 2),
                "median": round(median, 2),
                "variance": round(variance, 2)
            }

        except ZeroDivisionError:
            print("Cannot perform analysis: No data points available.")
    def get_results(self):

        return self.__analysis_results
```

Question-3 - Testing your code (5 pts) Write at least 5 test cases to test different parts of your implementation. We have provided an example. You will get full points as long as you have 5 test cases and a description of what is being tested. Please be sure to include at lease 2 test cases for perform_analyis method which should include at least one test case for checking exceptions. You are welcome to write your test cases using python unit test library. However, it is not required.

[11]:
```
#Example 1- Test Student Class constructor with student_id 1001, major "Data␣
↪Science, and university Harvard"
```

```
john = Student(1001, "Data Science", "Harvard")
print(f"Student id is : {john.student_id}\nMajor: {john.major}\nUniversity:
  ↪{john.university}")
```

```
Student id is : 1001
Major: Data Science
University:Harvard
```

```
[16]: #Example 2- Test Project Class constructor with student_id 1001, major "Data␣
        ↪Science, university Harvard, project_id 1,
      #data_points [65,75,95,99]"

      my_project = Project(1001, "Data Science", "Harvard", 1, [65,75,95,99])
      print(f"Student id is : {my_project.student_id}\nMajor: {my_project.
        ↪major}\nUniversity:{my_project.university}\n\
      project_id:{my_project.project_id}\ndata_points : {my_project.
        ↪data_points}\nis_active:{my_project.is_active()}" )
```

```
Student id is : 1001
Major: Data Science
University:Harvard
project_id:1
data_points : [65, 75, 95, 99]
is_active:True
```

```
[18]: #Test that a Project object can be instantiated with valid input values
      my_project2 = Project(101, "Data Science", "University of Illinois", 501, [1,␣
        ↪2, 3])
      print("Test Case 3 Passed: Project object created successfully.")
```

```
Test Case 3 Passed: Project object created successfully.
```

```
[19]: #Test that numeric data can be added successfully
      my_project2.add_data([10, 20, 30])
      print("Test Case 4 Passed: Data added:", my_project2.data_points)
```

```
Test Case 4 Passed: Data added: [1, 2, 3, 10.0, 20.0, 30.0]
```

```
[20]: #test that add_data raises an error when given non-numeric input
      try:
          my_project2.add_data(["bad", 22])
          print("Test Case 5 Failed: No error raised.")
      except TypeError:
          print("Test Case 5 Passed: TypeError raised for invalid data.")
```

```
Test Case 5 Passed: TypeError raised for invalid data.
```

```
[21]: #test that perform_analysis() works and stores results correctly
      my_project2.perform_analysis()
```

```
print("Test Case 6 Passed: Analysis results:", my_project2.
↪_Project__analysis_results)
```

```
Test Case 6 Passed: Analysis results: {'mean': 11.0, 'median': 6.5, 'variance':
137.6}
```

[ ]:
```
#test that perform_analysis() handles empty data
my_project3 = Project(2, "Math", "University of Kansas", 109, [1, 2, 3])
my_project3.perform_analysis()
print("Test Case 7 Passed: No crash on empty data.")
```

```
Test Case 7 Passed: No crash on empty data.
```

0.1.5 Question 4-9 (30 points)

Absolute Tests - For Question 4-9 simply run the cell below. We will verify your output with expected outputs. You implementation should be based on the requirement and not these test cases. If you have implemented the requirement correctly, your output should be as expected. Your implementation should be correct to get full credit here. Please do not modify the code for question 4-9

0.1.6 Question 4: Tests Class and Objects are created, and perform_analysis yields results as expected. (10 pts)

[27]:
```
test1 = [(1000, 'Computer Science', 'HARVARD', 1, [65,75,95,99]),(1001,␣
↪'Computer Science', 'MIT', 2, [95,35,75,90,91]),\
         (1003,'Data Science', 'Cornell', 3, [75,85,95,99,33])]


for test in test1:
    project = Project(test[0],test[1],test[2],test[3],test[4])
    project.perform_analysis()
    project_info = f"Student id : {project.student_id}\nMajor: {project.
↪major}\nUniversity:{project.university}\n\
project_id:{project.project_id}\ndata_points : {project.data_points}\nis_active:
↪{project.is_active()}"
    print(project_info)
    print(f"analysis results :{project.get_results()}\n")
```

```
Student id : 1000
Major: Computer Science
University:HARVARD
project_id:1
data_points : [65, 75, 95, 99]
is_active:True
analysis results :{'mean': 83.5, 'median': 85.0, 'variance': 262.33}

Student id : 1001
Major: Computer Science
```

```
University:MIT
project_id:2
data_points : [95, 35, 75, 90, 91]
is_active:True
analysis results :{'mean': 77.2, 'median': 90.0, 'variance': 614.2}

Student id : 1003
Major: Data Science
University:Cornell
project_id:3
data_points : [75, 85, 95, 99, 33]
is_active:True
analysis results :{'mean': 77.4, 'median': 85.0, 'variance': 702.8}
```

[ ]:

0.1.7 Question 5: Tests if data points can be added and data statistics are updated.(5 points)

[28]:
```
test = (1000, 'Computer Science', 'HARVARD', 1, [65,75,95,99])
project = Project(test[0],test[1],test[2],test[3],test[4])
project.perform_analysis()
print(f"analysis results before : {project.get_results()}")
project.add_data(22)
project.perform_analysis()
print(f"analysis results after: {project.get_results()}")
```

```
analysis results before : {'mean': 83.5, 'median': 85.0, 'variance': 262.33}
analysis results after: {'mean': 71.2, 'median': 75.0, 'variance': 953.2}
```

[ ]:

0.1.8 Question 6: Tests if active can be set (5 points)

[29]:
```
test = (1000, 'Computer Science', 'HARVARD', 1, [65,75,95,99])
project = Project(test[0],test[1],test[2],test[3],test[4])
print(f"value of active at initialization: {project.is_active()}")
project.set_active(False)
print(f"value of active after calling set method: {project.is_active()}")
```

```
value of active at initialization: True
value of active after calling set method: False
```

[ ]:

0.1.9 Question 7: Tests if perform_analysis can handle exceptions for empty list (4 points)

```
[30]: test1 = (1000, 'Computer Science', 'HARVARD', 1, [])

      print(f"Test #1 : Test perform analysis if the data points is an empty list.")
      project = Project(test1[0],test1[1],test1[2],test1[3],test1[4])
      project.perform_analysis()
```

```
Test #1 : Test perform analysis if the data points is an empty list.
Cannot perform analysis: No data points available.
```

0.1.10 Question 8: Tests if constructor raises an error if non-numerical values are present in data_points (3 pts)

```
[31]: test2 = (1001, 'Computer Science', 'MIT', 2, [95,35,75,90,'a'])
      #print(f"\nTest #3 : Test constructor if data points has non-numerical values.")
      project = Project(test2[0],test2[1],test2[2],test2[3],test2[4])
```

0.1.11 Question 9: Tests if adding non-numerical values to data_points raises an error (3 pts)

```
[32]: test3 = (1000, 'Computer Science', 'HARVARD', 3, [100,100,100])
      project = Project(test3[0],test3[1],test3[2],test3[3],test3[4])
      project.add_data('xyz')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[32], line 3
      1 test3 = (1000, 'Computer Science', 'HARVARD', 3, [100,100,100])
      2 project = Project(test3[0],test3[1],test3[2],test3[3],test3[4])
----> 3 project.add_data('xyz')

Cell In[26], line 24, in Project.add_data(self, new_data)
     21         raise TypeError("All elements in the list must be numeric.")
     23 else:
---> 24     raise TypeError("Input must be a number or a list of numbers.")

TypeError: Input must be a number or a list of numbers.
```

```
[ ]:
```

```
[ ]:
```

0.1.12 Question 10 (5 pts)

Working with Class and objects is very important in data science You will be working with a lot of python libraries which are all created using class and objects. You have already worked with pandas. Let's tie what you have learned in this homework with what you have already learned. Run the cell below, and answer the question that follow

[33]:
```
import pandas as pd
midterm_scores = [96,90,85]
df = pd.DataFrame(midterm_scores, columns=['midterm_scores'])

shape = df.shape
print(shape)
df.sort_values(by='midterm_scores', ascending=False)
```

```
(3, 1)
```

[33]:
```
   midterm_scores
0              96
1              90
2              85
```

1. Identify the class name and the parameters which is being used to create the object df. Please refer to pandas documentation to answer this question. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html (2pt)
2. Identify the attribute being called on the object df. (1pt)
3. Identify the method, and the method parameters being called by our object df. (2 pt)

your answer here

1. class name is pandas.dataframe and the parameters are midterm_scores and columns.

2. the attribure is shape

3. the method is sort_values and the method parameters are by='midterm_scores' and ascending=False

0.1.13 Coding style (5 pts)

Please make sure your code follows the coding style as defined here.

1. Comments on functions.
2. Additional comments on parts of code that maybe difficult to understand
3. Remove any commented code i.e. your solution should only include the code required by the assignment. All experiemental code needs to removed on the submitted file.
4. Code Readibility (if part of your code is long, use multiple lines)

0.2 Submission on Gradescope

On canvas left menu -> click on Gradescope

Submit the jupyter notebook, and a pdf version of this notebook.

To create a pdf of this notebook : In your browser open print, and save as pdf. Name the pdf LastNameFirstName__pset7.pdf example: DoeJohn__pset7.pdf

Name this jupyter notebook with the same format LastNameFirstName.ipynb

Make sure that your notebook has been run before creating pdf. Any outputs from running the code needs to be clearly visible. We need both .ipynb, and pdf of this notebook to assign you grades.

Drop all the files in gradescope under PSET 7: Python OO Programming Exercise

```
[ ]:
```