COURSE WORK I: PROGRAMMING FOR

DATA SCIENCE

MSc Data Science: Coventry University UK (**2024.2 BATCH**)

Name: P D Lolitha Lakshan Weerasinghe

Index Number: COMScDS242P-003

**QUESTION 1**

## a)

* + Implemented "Student", "Teacher," and "Staff" classes derived from the parent "Person" class.
  + Defined "name," "age," and "address" in the "Person" class.
  + Created an "address" class which encapsulates attributes such as street, city, state and zip code. Added “Address” to the "Person" class which act as a compound attribute.
  + Also, initializers are created for the classes
  + Class and methods comments added.

## b)

* + Defined “assign\_grades” method in the "Student" class where the method accepts “subject\_grade\_dict “as a parameter which holds the subjects and the marks.
  + The method first checks for an empty dictionary, if not empty then proceeds with the calculation.
  + The method outputs the average grade.
  + Method comments are included.
  + Unit tests are added to verify the functionality of the method for the scenarios such as validity of the calculation, handling empty dictionary and how method behaves for a single value.

## c)

* + Added the “ssn" attribute to the "Person" class.
  + Created a getter and setter to access the “ssn" attribute, however on “Person” class this would raise an “AttributeError”.
  + This would prevent from any subclasses setting / accessing the “ssn” property inherently, if any customized implementation is required for “Student” and “Teacher”; A customized implementation is provided to set and return the “ssn”.
  + “ssn” is exposed through the initializer as well, therefore only way to set the “ssn” is through the setter.
  + Also string validation is done in the setter to only accepts valid “ssn”.
  + Further improvements can be done to incorporate regex to validate “ssn”.

## d)

* + Implemented “role\_duties(self)” method in the “Person” class which returns all the duties supported as an array. So multiple duties are supported by each “Person”.
  + The duties are declared as an enum.
  + Each sub class overrides “role\_duties(self)” method and returns suitable duty for each subclass.

## e)

* + A “Subject” class is declared to hold subject related information such as identifier and name.
  + A “ScheduleClass” class declared to hold class information such as “Subject”, day of the week, start time and end time.
  + The day of the week is declared as an enum with days of the week.
  + A “Subjects” array is declared in the subclass “Teacher” to hold the list of subjects taught by a “Teacher”.
  + The “Subjects” can be assigned to a “Teacher” by using the “add\_subject()” method in the “Teacher” subclass.
  + A “class\_schedule” dictionary is declared “Teacher” subclass to hold class schedule information.
  + Day of the week will act as the key in the dictionary and values will be a list of the class schedules.
  + A class schedule can be added using by introduced “schedule\_classes” method which checks the day exists on the dictionary, if exists appends the schedule else adds a new list with the schedule.

## f)

* + To keep track of student attendance a “AttendanceRecord” class is declared. This class holds information about “Subject”, attendance status and date.
  + A list named “attendance\_records” is declared in the “Student” class which holds the “AttendanceRecord” instances.
  + An “attendance(self, date: datetime, subject: Subject, attended: bool)” method creates a new “AttendanceRecord” and adds to the “attendance\_records” list.
  + display\_attendance(self) method iterates through “attendance\_records” list and calculates numbers of attended classes, number of missed classes and attended percentage.

## g)

* + Declared “StaffRole” enum which declares staff roles such as MANAGER, ADMINISTRATIVE and OTHER along with their base salary.
  + Modified Staff subclass with new attributes “role”, “years\_of\_service” and “salary”.
  + The “role” and “years\_of\_service” is initialized upon instance creation using the constructor and salary is initially initialized to the value of None.
  + The “calculate\_salary(self)” method will calculate the Staff instance salary using the “role” and “years\_of\_service”, then updates “salary” variable.
  + The “get\_salary()” will return the calculated salary , if the salary is not present then salary is calculated prior to returning the value.
  + Also, unit tests are added to verify calculations are correct for each role.

## h)

* + The “Person” super class is declared with two new methods get\_role (self) and display\_info(self).
  + The get\_role (self) method returns the role for the class, subclasses such as “Teacher” , “Student” and “Staff” overrides this method to return their own “role”.
  + The display\_info(self) method prints the name, age and role for the class and declared in the super class.
  + A custom implementation for the “The display\_info(self) method is provided in the ““Staff” subclass.
  + This custom implementation prints the number of years of service apart from the name ,age and role which is specific to the “Staff” class implementation.
  + Polymorphism is demonstrated by creating instances of the “Student” , “Teacher ” and “Staff” classes , added to a list and invoking the “display\_info” method in each instance which prints the class specific implantations.

