```python
"""
    15/11/22
    Kristina Kudryavtseva, kristina.kudryavtseva.001@student.uni.lu
    SPOP main file

    The program is an exercise platform for students.
    The program accomodates two types of users: schools, students.
    Schools set the study plan for each semester.
    Students solve the exercises, are evaluated and ranked based on their
solutions.

    Thsi file must be executed in order to run the program.
    This file contains the runAndDeploy() function needed to deploy and
run the system,
    as well as the student evaluation function evaluateStudent().

"""

from administrator import Administrator
from school import School
from semester import Exercise, Semester
from student import Student

userDatabase = {"students":{"Kristina":"pass123"},
                "schools":{"Awesome School":"pass234"},
                "admins":{"admin":"admin123"}}

defaultExercise = Exercise("Who is the Liskov substitution principle
named after?", "Barbara Liskov")
exerciseDatabase = [defaultExercise]

def login(database):

    username = input("Please enter your username:")
    password = input("Please enter password:")

    role = None

    for role in ["student", "school", "admin"]:

        if username in database[role+"s"].keys():
            if database[role+"s"][username] == password:
                print("Welcome to your ",role," account. \n", "You are
loggen in as ", username)
            else:
                print("Wrong Password")
            return role, username

    print("No account with such username")
    return role, username

def logout(role, usename):
    print("You are logged out.")
    role, username = None, None
    return role, username
```

```
""" The function initializes a school instance,
    initializes the semester instance and a student instance,
    evaluates the student.
"""

def runAndDeploy():


    role, username = None, None
    while role == None or username == None:
        role, username = login(userDatabase)

    current_semester = None

    while role != "admin":
        role, username = logout(role, username)
        print("Please log in as ", "admin")
        while role == None or username == None:
           role, username = login(userDatabase)

    admin = Administrator(username)
    print("Please, initialize a new semester.")
    current_semester = admin.setStudyPlan(exerciseDatabase)

    while role != "student":
        role, username = logout(role, username)
        print("Please log in as ", "student")
        while role == None or username == None:
           role, username = login(userDatabase)

    student = Student(username)
    try:
        student.exercises = current_semester.exercises
    except:
        print("No semester in progress")

    student.study(current_semester.exercises[current_semester.current]) #
redo the implementation
    evaluateStudent(student)

    while role != "school":
        role, username = logout(role, username)
        print("Please log in as ", "school")
        while role == None or username == None:
           role, username = login(userDatabase)

    school = School(username)
    school.addExcercise()

    return

""" The function takes an instance of class Student as a parameter
    and returns the student's score based on their solutions to the
exercises."""
def evaluateStudent(student):
    studentRecord = student.progress

    for exercise in studentRecord.keys():
        studentAnswer = studentRecord[exercise]
        if isinstance(studentAnswer, str):
            if exercise.solution == studentAnswer:
                studentRecord[exercise] = 1
            else:
                studentRecord[exercise] = 0

    student.score = sum(studentRecord.values())

    print(student.name, "'s score is ", student.score)

    return

# execution of the program
runAndDeploy()

from semester import Semester, Day, Exercise
from random import choice
# semester has a state - > calendar that shows what day it is
# program has a timer that counts days in the calendar

class Administrator():

    def __init__(self, id):
        self.id = id

    def setStudyPlan(self, exerciseDatabase): #database

        print("You are setting a study plan for this semester.\n")
        semesterLength = self.setNumberOfExercises()
        exercises = [None]
        for i in range(semesterLength):
            exercises.append(choice(exerciseDatabase))
        calendar = (None, [])

        """
        for i in range(1, semesterLength):
        # add a node to the linked list
            i+=1
        """

        print("Semester with ", semesterLength, " exercises
initialized!\n")

        semester = Semester(semesterLength, exercises)

        return semester


    """The function expects an integer input and returns this integer."""
    def setNumberOfExercises(self):

        n = int(input("Enter the number of exercises for this semester:
```

```
"))

        return n

"""
    15/11/22
    Kristina Kudryavtseva, kristina.kudryavtseva.001@student.uni.lu
    SPOP main file

    The program is an exercise platform for students.
    The program accomodates two types of users: schools, students.
    Schools set the study plan for each semester.
    Students solve the exercises, are evaluated and ranked based on their
solutions.

    This file contains the School class and corresponding functions,
    which requires user input from the school actor.

"""

from semester import Semester, Exercise

class School:
    """user_goal
       reuse: setNumberOfExercises(), addExercise()
    """
    def __init__(self, name, id=None):
      self.name = name
    #  self.id = id

    """The fucntion expects two strings that correspond to an exercise
task and a solution."""
    def addExcercise(self):

        print("~~~You are adding a new exercise~~~")
        task = input("Please enter the task:")
        solution = input("Please enter the solution:")
        print("\n")
        exercise = Exercise(task, solution, self.id)

        return exercise

"""
    15/11/22
    Kristina Kudryavtseva, kristina.kudryavtseva.001@student.uni.lu
    SPOP main file

    The program is an exercise platform for students.
    The program accomodates two types of users: schools, students.
    Schools set the study plan for each semester.
    Students solve the exercises, are evaluated and ranked based on their
solutions.

    This file contains the Student class and corresponding functions,
    which expect user input from the student actor.
```

```
"""

from school import School

class Student:

# we have to reinitialize a student when a new semester starts
# Student != Semester

# Progress: contains a pointer to the current day in a semester and a
list of ex+student solutions
    def __init__(self, name, id=None):
        self.name = name
       # self.id = id
        self.exercises = {None:None}
       # self.progress = [] # semester, semester day, completed exercises
with a score

# Possibly useless

    def study(self, exercise):

        print(self.name, "is solving exercises")

        self.openExercise(exercise)
        studentSolution = self.enterSolution()
        # self.exercises is a list but should be a dict
        self.exercises[exercise] = studentSolution # We could evaluate
automatically and store the score


    # prints exercise task
    def openExercise(self, exercise):
        print(exercise.question, " ?")
        return exercise.question

    def enterSolution(self):
        solution = input("Please enter your solution:")
        return solution

"""
    15/11/22
    Kristina Kudryavtseva, kristina.kudryavtseva.001@student.uni.lu
    SPOP main file

    The program is an exercise platform for students.
    The program accomodates two types of users: schools, students.
    Schools set the study plan for each semester.
    Students solve the exercises, are evaluated and ranked based on their
solutions.

    This file contains the Semester class.

"""

# does a Semester instance depend on a student or vice versa?
# a pointer to the current day - in Semester or in Student
class Semester():
    def __init__(self, length, exercises):
        self.length = length
        self.exercises = exercises # calendar is a tuple: a linked list
of days, a pointer
        self.current = 1

class Exercise():
    def __init__(self, question, solution, schoolId="test"): # add
exercise ID
        self.question = question
        self.solution = solution
        self.level = schoolId


class Day(): # node
    def __init__(self, id, exercise):
        self.id = id
        self.exercise = exercise
        self.next = None # last day in the calendar

class Calendar(): # linked list of Days
    def __init__(self, head = None):
        self.head = head
        self.count = 0
```