MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG445 Software Development with Scripting Languages - Midterm Exam
Fall 2024/2025

Assignment 1: A Data Analyser for University Resources

This assignment aims to help you practice Python basics with object-oriented programming including primitive types, variables, operators, decision making, loops, sequences, dictionaries, functions, modules, regular expressions, exception handling, command line arguments, and file processing. Your main task in this assignment is to develop an application for analysing the allocation of university resources across different departments.

Overview

Universities employ many technical systems to manage their day-to-day operations. Many universities are interested in analysing the data they collect to direct their future efforts. One direction they would like to explore is expanding the departments which are in demand. A university has tasked you with developing an application to assist them in analysing the data to find the trends within it.

The application you will develop will process two **txt** data files[1]. You will be provided with two input files:

1. **departments.txt** that contains information regarding the departments.
2. **courses.txt** that contains information regarding the sections of courses.

Sample data for the departments.txt file will look like:

```
Department Code,Department Short Name,Department Name
355,CNG,Computer Engineering
356,EEE,Electrical and Electronics Engineering
365,MECH,Mechanical Engineering
```

Sample data for courses.txt will look like:

```
Department Code,Course Code,Course Name,Instructor,Section,Capacity,Registered
Students
355,CNG 100,INTRODUCTION TO INFORMATION TECH. AND APPLICATIONS,HUSEYIN
SEVAY,S1,106,100
355,CNG 100,INTRODUCTION TO INFORMATION TECH. AND APPLICATIONS,MUHAMMED AKIF
AGCA,S2,104,97
355,CNG 100,INTRODUCTION TO INFORMATION TECH. AND APPLICATIONS,MUHAMMED AKIF
AGCA,S3,109,108
355,CNG 101,COMPUTER ENGINEERING ORIENTATION,YELIZ YESILADA,S1,108,102
```

Each line contains information for one section. Lecture sections will be written as Sx while lab sections will be written as Labx (where x is a number). As can be seen, the data will be separated with commas, thus the parsing must be done accordingly. To avoid unnecessary complexity, instructor names will only contain letters from the English Alphabet.

[1] These data files are prepared for this assignment, and they do not include real numbers.

Implementation Requirements

This application should receive the data file names as a command-line argument and then create the appropriate object structures. Please note that the names and content of the data files can be changed but their formats will be the same.

You need to name your main program as **universityanalyser.py**. The first argument should be the departments file, and the second argument is the courses file. The sample command-line execution of the program should be as follows.:

```
python universityanalyser.py departments.txt courses.txt
```

The required object class structure is presented in Fig. 1.

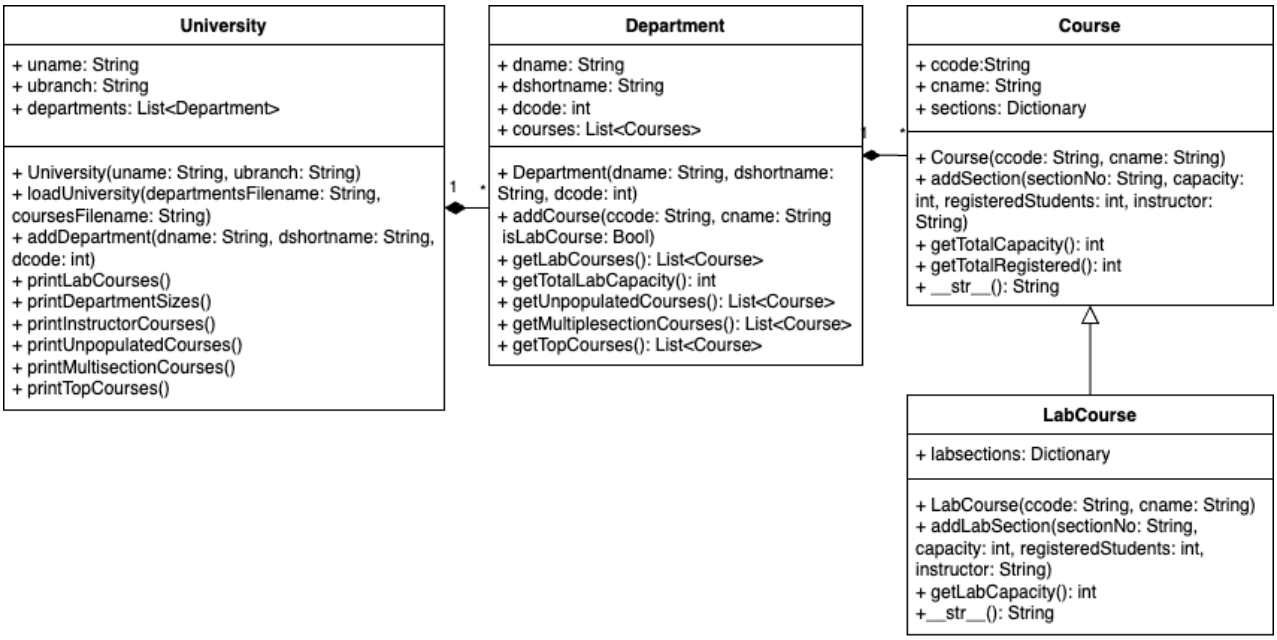

Figure 1: The Class Diagram of System

You are free to add helper functions to all objects to maintain code cleanliness. The class specifications are given below.

University Class

Attribute	Explanation
uname	University name, such as "METU"
ubranch	University branch, such as "NCC"
departments	List that holds the departments of the university as Department objects

Method	Explanation
University	Constructor that takes university name and university branch to create a University object

loadUniversity	Create the object structure by reading the given files whose format is given above. Please note that course code should start with the short department name and then followed by three or four digits (such as CNG445). If not, the course in the file should be skipped. **You need to use a regular expression for this purpose.**
addDepartment	Create and add a department to the university by taking department name, short name, and code
PrintLabCourses	Prints all the details of the courses that offer labs in each department
PrintDepartmentSizes	Display a pie chart of all departments with respect to the total number of registered students of all courses (excluding the capacity of the lab sections, if any).
PrintInstructorCourses	Ask the user for the name of an instructor then print the details of the course that they teach. If the instructor teaches multiple courses, the method will list all the courses and ask the user which course they would like to print then print its details. If the instructor is not found, print the appropriate message.
PrintUnpopulatedCourses	Print the details of all courses which have less than 5 registered students (excluding the number of registered students in the lab sections, if any) in each department. If none exist in a department, print an appropriate message.
PrintMultisectionCourses	Print the details of all courses which have more than 1 section or more than 1 lab section in each department. If none exist in a department, print an appropriate message.
PrintTopCourses	Print the details of the course with the highest number of registered students in each department (excluding the number of registered students in the lab sections, if any). If there are other courses with the same number of students, print them too.

Department Class

Attribute	Explanation
dcode	Department code, such as 355
dshortname	Department short name, such as CNG
dname	Department name, such as Computer Engineering
courses	List that holds the courses of the department as Course objects

Method	Explanation
Department	Constructor that takes department code (default value: 0), department short name (default value: ?) and department name (default name: ?) to create a Department object
addCourse	Create and add a Course or LabCourse to the department using course code, course name
GetLabCourses	Return the list of courses that offer labs
GetTotalLabCapacity	Return the total capacity of all lab sections under the department
GetUnpopulatedCourses	Return a list of all courses in the department which have less than 5 registered students (excluding the number of number of students in the lab sections, if any). May return an empty list.

GetMultisectionCourses	Return a list of all courses in the department which have more than 1 section or more than 1 lab section. May return an empty list.
GetTopCourses	Return a list with the courses in that department which have the highest number of students (excluding the number of number of students in the lab sections, if any). May return a list with 1 item.

Course Class

When a course does not have any lab section, it should be created as a Course.

Attribute	Explanation
ccode	Course code, such as "CNG445"
cname	Course name, such as "Software Development with Scripting Languages"
sections	Dictionary that keeps tracks of the section details – Dictionary key will be a section number (such as, S1), and for each section, the capacity, the number of registered students, and the instructor's name should be stored in a tuple format such as ("YELIZ YESILADA", 108, 102).

Method	Explanation
Course	Constructor that takes course code and course name to create a Course object
addSection	Take section number, capacity, number of registered students, and instructor's name, and add it to the dictionary
GetTotalCapacity	Return the total capacity of all sections of the course.
GetTotalRegistered	Return the total number of registered students of all sections of the course.
__str__	Allow to print a Course object with course code, course name, and section details

LabCourse Class

Inherits from Course Class. When a course is offered with labs, then it should be created as a LabCourse

Attribute	Explanation
labsections	Dictionary that keeps tracks of the lab section details – Dictionary key will be a lab section number (such as, Lab1), and for each section, the capacity, the number of registered students, and the instructor's name should be stored ("Asst. Cng 1", 40, 38).

Method	Explanation
LabCourse	Constructor that takes course code and course name to create a LabCourse object
AddLabSection	Take lab section number, capacity, number of registered students, and instructor, and add it to the dictionary
getLabCapacity	Returns the total capacity of all the lab sections offered by the course.
__str__	Allow to print a LabCourse object with course code, course name, and section and lab section details

Main Program Operation

On startup, the application should take the filenames as command-line arguments. The application should then instantiate a university with name "METU" and branch "NCC" and read the given files to

populate the rest of the objects accordingly by using the appropriate methods. Then, the application should show the following menu in a loop:

1. Print all the lab courses in the university: The application should print the department name, and the total capacity of all the lab sections of the department, followed by a list of all courses which have a lab component. You should mention course code, course name, and section and lab details for each course.
2. Print department sizes: the application should display a pie chart showing the total capacity of all courses offered by the departments (excluding the capacity of the lab sections, if any). You only need to print the names of the departments on the pie chart slices. You may follow the sample code given at the end of this document.
3. Print instructor courses: the application should ask the user for the name of an instructor. Then it will print the course in which they teach a section. If the instructor teaches sections in multiple courses, the application should ask which course the user would like to read the details of and print the details of that course. If the instructor's name is not found, an appropriate message should be printed. You may assume that there will be no errors in input and the instructor's name will be entered exactly as the name in the file.
4. Print unpopulated courses: the application should print the details of all courses which have less than 5 students registered to it (excluding the number of registered students in the lab sections, if any) in each department. If none exist in a department, an appropriate message should be printed.
5. Print multi-section courses: the application should print the details of all courses which have more than 1 lecture section or 1 lab section in each department. To clarify, a course should not be printed if it has 1 lecture section and no lab sections or 1 lecture section and 1 lab section. If none exist in a department, an appropriate message should be printed.
6. Print top course(s): the application should print the course with the highest number of registered students in each department (excluding the number of registered students in the lab sections, if any). If other courses exist which have the same number of registered students, they should be printed as well.

Printing Charts

Provided below is a sample code to print a pie chart. Please note that you need to install matplotlib beforehand.

```
import matplotlib.pyplot as plt
departments = ['CNG', 'EEE', 'MECH']
sizes = [891, 572, 312]
plt.pie(sizes, labels=departments)
plt.title('Department Sizes')
plt.show()
```

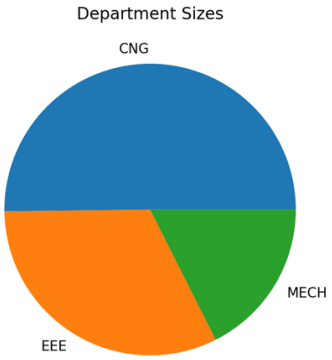

Figure 2. A pie chart

Rules

- You need to write your program by using **Python 3.x**.
- You can **only** use all built-in functions and modules (apart from matplotlib)
- You also need to create a file called ReadMe.txt which contains the following items. Please note that **if you do not submit ReadMe.txt, your submission will not be evaluated**.
  - Team members
  - Which version of Python 3.x you have used
  - Which operating system you have used
  - How you have worked as a team, especially how you have divided the tasks among the team members (who was responsible for what?), how you have communicated, how you have tested the program, etc.
- You need to put all your files into a folder which is named with your student id(s) and submit the compressed version of the folder in the .zip format.
- **Only one team member** should submit the assignment.
- **Code quality, modularity, efficiency, maintainability, and appropriate comments** will be part of the grading.

Grading Policy

Grading Item	**Mark(out of 100)**
Operations in main (reading command-line arguments and instantiating the university object)	5
University class	40
Department Class	25
Course Class	20
LabCourse Class	10