Recall the data structure that you designed last week for storing grades of lab groups. In this example class, we are about to develop a grading system consisting of the following functionalities:

* Input individual grades, e.g., the grade of student no. 1 in FE1 is 90.
* Query the grade of a student in a lab group
* List all the grades in a lab group
* Get the highest grade in a lab group

**Discussion 1**

Develop a Python function inputRecord(dataBase, group, id, score) for TAs to enter one record, where dataBase is the database implemented by your data structure, group is a string representing a group name, id is a student’s id number (positive integers ranging from 1 to 40), and score is the grade of the student.

**Discussion 2**

Develop a Python function query(dataBase, group, id) for TAs to get the score of a student in a lab group, where dataBase is the database implemented by your data structure, group is a string representing a group name, and id is the student’s id. This function should return the score.

**Discussion 3**

Develop a Python function listGrades(dataBase, group) for TAs to get all the student grades in a group, where dataBase is the database implemented by your data structure and group is a string representing a group name. This function should return a list of grades in the group.

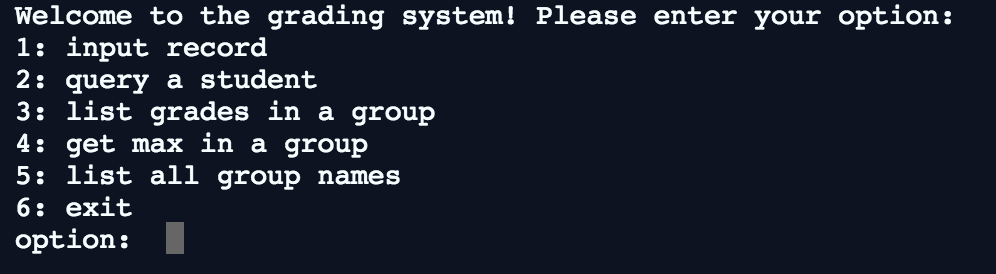
**Discussion 4**

Develop a Python function maxGrade(dataBase, group) for TAs to get the highest grade in a group, where dataBase is the database implemented by your data structure and group is a string representing a group name. This function should return the highest grade.

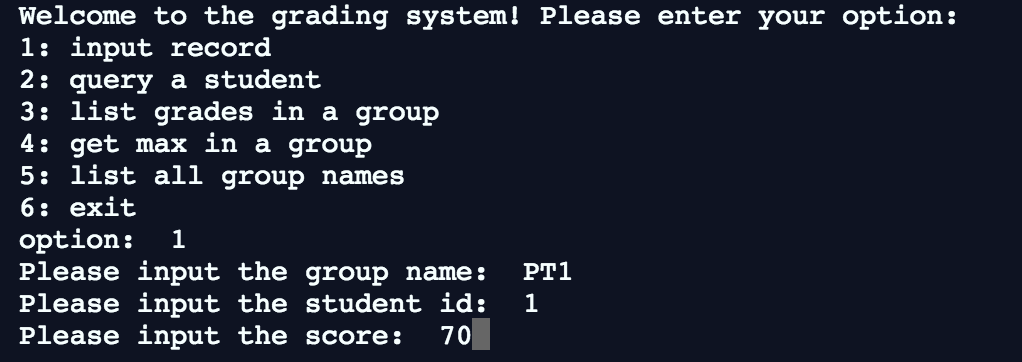
**Discussion 5 (Optional)**

Develop the main program to show the options to users, get the option, and invoke the corresponding functions developed.

For example, show the available options and get the option from users:



For example, invoke the corresponding function:



#dataBase

labClass = {}

#Input student's score into dataBase

def inputRecord(dataBase, group, ID, score):

    if group not in dataBase:

        dataBase[group] = {}

    dataBase[group][ID] = score

    print("Student's score has been updated.")

#Calls for student's score from dataBase

def query(dataBase, group, ID):

    try:

        grade = dataBase[group][ID]

        print(f"This student's score is {grade}.")

    except KeyError:

        print("Student's score not in data base.")

#List every student's score in a group

def listGrades(dataBase, group):

    try:

        for ID in dataBase[group]:

            print(f"{ID}: {dataBase[group][ID]}")

    except KeyError:

        print("Group not in data base.")

#List max grade in a group

def maxGrade(dataBase, group):

    try:

        max\_grade = max(dataBase[group].values())

        print(f"Max grade in {group} is {max\_grade}.")

    except:

        print("Group not in data base.")

#List every group in a dataBase

def listGroups(dataBase):

    if len(dataBase) == 0:

        print("Data base is empty.")

    else:

        i = 1

        for group in dataBase:

            print(f"{i}. {group}")

            i += 1

#Prompts action from user

def main\_menu():

    while True:

        try:

            print("\n\

Welcome to the grading system! Please enter your option: \n\

1: Input record\n\

2: Query a student\n\

3: List grades in a group\n\

4: Get max grade in a group\n\

5: List all group names\n\

6: Exit")

            option = input("Option: ")

            if float(option) % int(option) != 0:

                print("Invalid input! Please enter 1-6.")

                continue

            break

        except:

            print("Invalid input! Please enter 1-6.")

    return int(option)

#ID only takes 1-40 as inputs

def i\_d():

    while True:

        try:

            x = input("Please input the student id: ")

            if float(x) % int(x) != 0 or int(x) < 1 or int(x) > 40:

                print("Invalid input! Student's ID can only be 1-40.")

                continue

            break

        except:

            print("Invalid input! Student's ID can only be 1-40.")

    return x

def my\_score():

    while True:

        try:

            y = input("Please input the score: ")

            if float(y) % int(y) != 0 or int(y) < 1 or int(y) > 100:

                print("Invalid input! Student's ID can only be 1-100.")

                continue

            break

        except:

            print("Invalid input! Student's ID can only be 1-100.")

    return y

while True:

    option = main\_menu()

    if option == 1:

        group = input("Please input the group name: ")

        ID = i\_d()

        score = my\_score()

        inputRecord(labClass, group, ID, score)

    elif option == 2:

        group = input("Please input the group name: ")

        ID = i\_d()

        query(labClass, group, ID)

    elif option == 3:

        group = input("Please input the group name: ")

        listGrades(labClass, group)

    elif option == 4:

        group = input("Please input the group name: ")

        maxGrade(labClass, group)

    elif option == 5:

        listGroups(labClass)

    elif option == 6:

        print("Program exited.")

        quit()