Assignment 9

ELP 780 Software Lab

Varun Sood

2017EET2839

2017-19

A report for the assignment on

Python Programs(oops)

Bharti School of
Telecommunication Technology and Management
IIT DELHI
India

August 28, 2019

Contents

1 Problem Statement 1

1.1 Problem statement

Problem Statement Develop a Department management system (DMS). In this system, the three entities are Department, Employee and Manager. The DMS should be able to handle many department which can be joined by many employee(upto max value given for each department) by a single manager. **Basic Requirements:** Employee can join a department and later also leave from the department. Each department must be assigned a manager. Ability to modify any detail of the department. Employee will have the following attribute and functions

- Employee can join a department and later also leave from the department.
- Each department must be assigned a manager.
- Ability to modify any detail of the department.
- Employee will have the following attribute and functions

  Name : String Unique
  EmpId : String Unique
  Designation : String Trainee Engineer, Software Engineer and System Analyst
  Experience year : integer 1,2,3,4,5
  Department joined : department name
  addEmployee : to add the employee details
  join(department) : function to join a department
  leave(department) : function to leave from a department already taken
  showEmployee : Display all details of the employee

- Manager will have the following attribute and functions

  name : String Unique
  position : String Project Manager, Delivery Manager
  addManager : to add the manager details
  departmentAssigned : list of department
  showManager : Display all details of the manager

- Department will have the following attribute and function

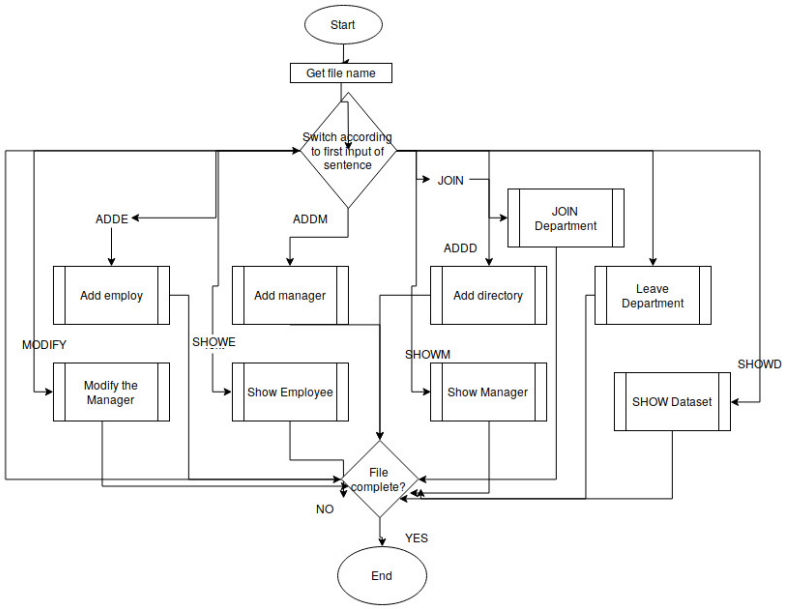
name : String Unique
maxEmployee : maximum number of employee that can join
manager : Manager assigned
empJoined : list of employee joined the department
addDepartment : to add the department details
Modify(max-emp, manager) : function to modify the max employee limit and the manager of the department
showDepartment : Display everything of department

- Write Getter and Setter function in all classes

1.2 Assumptions

- The designation are assumed to be snake case.
- Input is taken from the command line.
- Inputs are assumed to be of proper length as mentioned in the sample file.

1.3 Program Structure



1.4 Algorithm and Implementation

- Input the file name.
- For each of the line parse and run according to the keyword.
- Run function.
- Check if the no of lines are complete If yes complete.

1.5 Input and Output Format

- **Input format**
  Using the command line or terminal get the filename.

- **Output format**
  Output according to the input file.

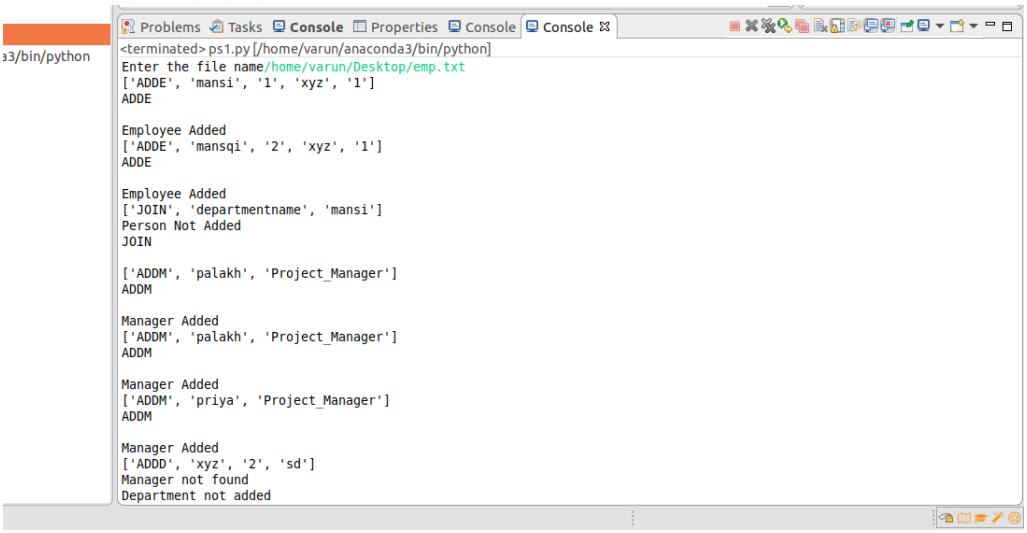
1.6 Test Cases

Input
Enter the file name along with the path
/home/varun/Desktop/emp.txt
Output
According to the input file , output will be corresponding to that.

1.7 Difficulty/Issues faced

- Eclipse was difficult to use
- Python oops is complex.
- latex is difficult to use

1.8 Screenshots



Eclipse Workstation

2 Appendix

2.1 Appendix-A : code for ps1

code1

```
%\lstinputlisting[basicstyle=\small,language=python]{ps1.py}
//==========================================================
// Name        : ps2.py
// Author      : Varun Sood
// Version     :  1
// Copyright   : Your copyright notice
//==========================================================
# program is working fine
# program is working fine
#class employ

class Employee():
    #list of employee
    List=[]
    #initialize
    def __init__(self):
        pass
    #add employee

    def addEmployee(self,name,ID,designation,experience):
        department='NULL'        #department
        self.L=[name,ID,designation,experience,department]  #create a
    list
        self.List+=[self.L]              #Add to main list
        print("Employee Added")
        #show employee
    def showEmployee(self,Employee_name):
        for i in self.List:
#     print required employee
            if i[0]==Employee_name:
                print(i[0],i[1],i[2],i[3],i[4])
                print("Employee Shown")
                break
            #if no employ is present
            print("NO employ")

        #Employ  exist or not
        #
    def exist(self,emp_name):
        for i in self.List:
            if i[0]==emp_name:
                return True
        return False

```

```
    # Add a department

    def add_department(self,emp_name,depart_name):
        for i in self.List:
            #check if employ exist or not
            if i[0]==emp_name:
                i[4]=depart_name

    # Remove a department from the employee
    def remove_department(self,emp_name,depart_name):
        for i in self.List:
            #check for user
            if i[0]==emp_name:
                i[4]='NULL'
#setter and getter format
            '''@property
    def name(self):
        return self.name

    @property
    def ID(self):
        return self.EmpID
    @property
    def department(self):
        return self.department
    @property
    def Designation(self):
        return self.designation
    @property
    def experience(self):
        return self.experience

    @ID.setter
    def ID(self,ID):
        self.EmpID = ID

    @department.setter
    def department(self,department):
        self.department = department
    @name.setter
    def name(self,name):
        self.name = name

    @Designation.setter
    def Designation(self,designation):
        self.Designation = designation

    @experience.setter
    def experience(self,experience):
        self.experience = experience
```

```
        '''
#Manager class
class Manager():
    M=[]
    #main list of manager
    #initial class
    def __init__(self):
        pass

    # Add manager to list
    def addManager(self,name,position):
        #CHECK if project manager
        if position =='Project_Manager':
            department='NULL'
            #create a dummy list
            self.m=[name, position,department]
            #add to main list
            self.M+=[self.m]
            #if delivery manager more than 1 department can be added
        elif position =='Delivery_Manager':
            department=[]
            self.m=[name, position,department]
            self.M+=[self.m]
        print("Manager Added")
        # show manager
    def showManager(self,manager_name):
        for i in self.M:
            #check manager name
            if i[0] ==manager_name:
                print(i[0],i[1],i[2])
                print("Manager Shown")
                break
        print("Manager not present")
        # find the manager name true if found
    def find(self,manager_name):
        for i in self.M:
            #check manager name
            if(i[0]==manager_name):
                return True
        return False

    #check for existence of manager name
    def exist(self,manager_name):
        for i in self.M:
            if i[0]==manager_name:
                return True
        return False
    #check for existence of manager name
    def exist_m(self,emp_name):
        for i in self.M:
```

```
            if i[0]==emp_name:
                #if project manager or not
                if i[1]=="Project_Manager" and i[2]=='NULL':
                    return True
                else:
                    print("Already Assignment")
                    #if delivery manager or not
                    if i[1]=="Delivery_Manager" :
                        return True
                    return False
        return False
    #setter and getter commented
    '''
    @set_position.setter
    def set_position(self,position):
        self.position = position

    @set_experience.setter
    def set_experience(self,experience):
        self.experience = experience
    @addEmployee.setter
    def addEmployee(self,name,ID,designation,department,experience):
        super(Employee, self).__init__(name)
        self.EmpID = ID
        self.Designation = designation
        self.department = department
        self.experience =self.experience

    @property
    def get_ID(self):
        return self.EmpID
    @property
    def get_department(self):
        return self.department
    @property
    def get_Designation(self):
        return self.designation
    @property
    def get_experience(self):
        return self.experience
    '''
    #department class
class Department:
    D=[]
    #department list
    def __init__(self):
        pass

    # add a department
    def addDepartment(self,name,maxEmployee,manager):
```

```
        empJoined=[]
        #if manager exist
        self.d=[name,maxEmployee,manager,empJoined]
        self.D+=[self.d]

        #maxchange change the max change
    def max_change(self,depart_name,max):
        for i in self.D:
            if i[0]==depart_name:
                i[1]=str(max)
                break

    #show department
    def showDepartment(self,depart_name):
        for i in self.D:
            if i[0]==depart_name:
                #check for the department
                print(i,end=" ")
                print("\nDepartment Shown")
        print("Department nort present")

    ## existence of the department
    def exist(self,depart_name):
        for i in self.D:
            if i[0]==depart_name:
                return True
        return False

    #max space check
    def space(self,depart_name):
        for i in self.D:
            if i[0]==depart_name:
                count=len(i[3])
                if count < int(i[1]):
                    return True
                else:
                    return False

    #add employ to department
    def add_emp(self,depart_name,empl_name):
        for i in self.D:
#             check for department
            if i[0]==depart_name:
                i[3].append(empl_name)
#                 print(type(i[3]))
                break

    # change the department
    def  change(self,depart_name,empl_name):
        for i in self.D:
```

```
            if i[0]==depart_name:
                i[2]=empl_name
                break

        #remove the employee
    def rem_emp(self,depart_name,empl_name):
        for i in self.D:
            if i[0]==depart_name:
                for emp in i[3]:
                    if emp==empl_name:
                        i[3].remove(empl_name)
                        return True
        print("Employee not Present")
        return False

    #not empty

    def not_empty(self,depart_name):
        for i in self.D:
            if i[0]==depart_name:
                count=len(i[3])#employ count
                if count != 0 :
                    return True
                else:
                    return False

filename = input("Enter the file name along with the path\n")
file = open(filename, "r")
#object of employee,manager,department
obj_emp = Employee()
obj_manag = Manager()
obj_depart = Department()
# get line from the file and parse the line
for line in file:
    list=line[:-1].split(" ")
    print(list)
    #Add employee
    if list[0] == 'ADDE':
        #adding employee
        print("ADDE\n")
        #add employee
        obj_emp.addEmployee(list[1],list[2],list[3],list[4])
    #Add manager
    elif list[0] == 'ADDM':
        #adding manager
        print("ADDM\n")
        #Add manager
        obj_manag.addManager(list[1],list[2])
    # add Department
    elif list[0] == 'ADDD':
```

```
        flag=False
        #find the managername
        if obj_manag.find(list[3])==True:
            flag=True
        else:
            flag=False
        if flag== True:
            # add Department
            obj_depart.addDepartment(list[1], list[2],list[3])
            print("Department added")
        else:
            print("Manager not found")
            print("Department not added")
        print("ADDD\n")

        #Join the department
    elif  list[0] == 'JOIN':
        #existance of department
        if obj_depart.exist(list[1])==True:

            if obj_depart.space(list[1]) ==True:
                print("space present")
                print("Person Added")
                if obj_emp.exist(list[2]):
                    print("Employee present")
                    #add employ
                    #add the department
                    obj_depart.add_emp(list[1],list[2])
                    obj_emp.add_department(list[2],list[1])#2 =person name 1= depart name
                else:
                    print("employee not present")
                    print("Person Not Added")
            else:
                print("space not present")
                print("Person Not Added")
        else :
            print("Person Not Added")

        print("JOIN\n")
        #leave the deprtment
    elif list[0] == 'LEAVE':
        #department exist

        if obj_depart.exist(list[1])==True:
            #not empty
            if obj_depart.not_empty(list[1]) ==True:
                print("space not empty")
                if obj_emp.exist(list[2]):
                    print("Employee present")
```

```
                    #remove employ
                    if obj_depart.rem_emp(list[1],list[2]) ==True:
                        #update department
                        obj_emp.remove_department(list[2],list[1])#2 = person name 1= depart name
                        print("Employee removed")
                    else:
                        print("Employee not removed")
                else:
                    print("employee not present")
                    print("Person Not Removed")
            else:
                print("space not present")
                print("Person Not Removed")
        else :
            print("Person Not Removed")
        print("LEAVE\n")
        #modify the manger / of department
    elif list[0] == 'MODIFY':
        if obj_depart.exist(list[1])==True:
            #check if manger is not assigned any department or is delivery manager
            if obj_manag.exist_m(list[3])==True:
                obj_depart.change(list[1],list[3])
                print("Manager Added")
                obj_depart.max_change(list[1],list[2])
            else:
                obj_depart.max_change(list[1],list[2])
                print("Manager doesnot exist")
        else:
            print("Department Doesnot exist")
        print("MODIFY\n")
    #show employee
    elif  list[0] == 'SHOWE':
        #show the employee if exist
        obj_emp.showEmployee(list[1])
        print("SHOWE\n")
    elif list[0] == 'SHOWM':
        #show the manager if exist
        obj_manag.showManager(list[1])
        print("SHOWM\n")
    elif  list[0] == 'SHOWD':
        #show the department if exist
        obj_depart.showDepartment(list[1])
        print("SHOWD\n")
    else :
        print("NOthing")
```

Listing 1: ps2.c

References

[1] Code Python from Scratch: Object Oriented Programming,
https://code.tutsplus.com/articles/python-from-scratch-object-oriented-programmin

[2] Youtube tutorial for python,
https://www.youtube.com/watch?v=ZDa-Z5JzLYM&list=
PL-osiE80TeTt2d9bfVyTiXJA-UTHn6WwU&index=34

[3] Object Oriented Programming,
http://anandology.com/python-practice-book/object_oriented_
programming.html

[4] Latex format guide,
https://www.sharelatex.com/