

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 1
 - 1.2 Assumptions 2
 - 1.3 Program Structure 2
 - 1.4 Algorithm and Implementation 3
 - 1.5 Input and Output Format 3
 - 1.6 Test Cases 3
 - 1.7 Difficulty/Issues faced 3
 - 1.8 Screenshots 4
- 2 Appendix 5**
 - 2.1 Appendix-A : code for ps1 5

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

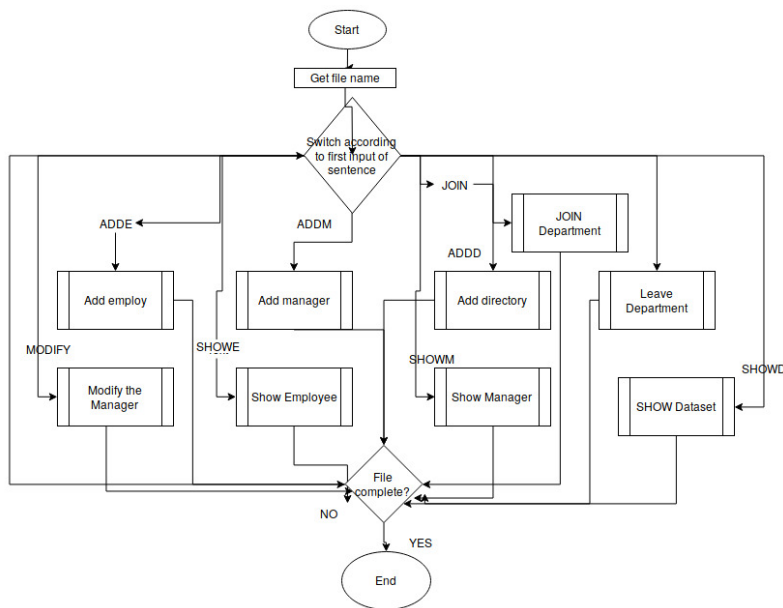
name : String Unique
 maxEmployee : maximum number of employee that can join
 manager : Manager assigned
 empJoined : list of employee joined the department
 addDepartment : to add the department details
 Modify(max-emp, manager) : function to modify the max employee limit
 and the manager of the department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

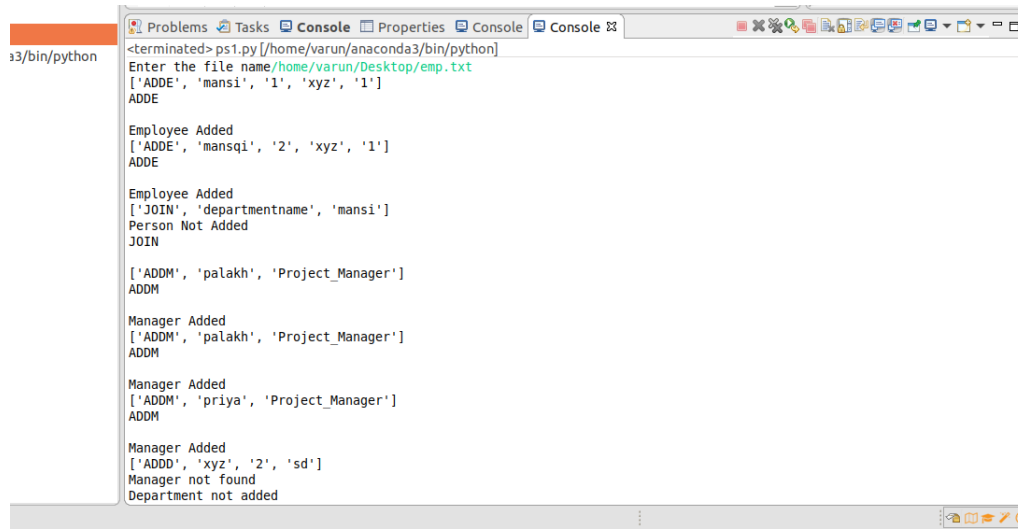
Output

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



```
<terminated> ps1.py [/home/varun/anaconda3/bin/python]
Enter the file name /home/varun/Desktop/emp.txt
['ADDE', 'mansi', '1', 'xyz', '1']
ADDE

Employee Added
['ADDE', 'mansiqi', '2', 'xyz', '1']
ADDE

Employee Added
['JOIN', 'departmentname', 'mansi']
Person Not Added
JOIN

['ADDM', 'palakh', 'Project_Manager']
ADDM

Manager Added
['ADDM', 'palakh', 'Project_Manager']
ADDM

Manager Added
['ADDM', 'priya', 'Project_Manager']
ADDM

Manager Added
['ADDD', 'xyz', '2', 'sd']
Manager not found
Department not added
```

Eclipse Workstation

2 Appendix

2.1 Appendix-A : code for ps1

code1

```
1 %\lstinputlisting[basicstyle=\small,language=python]{ps1.py}
2 //=====
3 // Name      : ps2.py
4 // Author    : Varun Sood
5 // Version   : 1
6 // Copyright : Your copyright notice
7 //=====
8 # program is working fine
9 # program is working fine
10 #class employ
11
12 class Employee():
13     #list of employee
14     List=[]
15     #initialize
16     def __init__(self):
17         pass
18     #add employee
19
20     def addEmployee(self,name,ID,designation ,experience):
21         department='NULL' #department
22         self.L=[name,ID,designation ,experience ,department] #create a
list
23         self.List+=self.L #Add to main list
24         print("Employee Added")
25         #show employee
26     def showEmployee(self ,Employee_name):
27         for i in self.List:
28 #         print required employee
29             if i[0]==Employee_name:
30                 print(i[0] , i[1] , i[2] , i[3] , i[4])
31                 print("Employee Shown")
32                 break
33             #if no employ is present
34             print("NO employ")
35
36         #Employ exist or not
37         #
38     def exist(self ,emp_name):
39         for i in self.List:
40             if i[0]==emp_name:
41                 return True
42         return False
43
```

```

44     # Add a department
45
46     def add_department(self, emp_name, depart_name):
47         for i in self.List:
48             #check if employ exist or not
49             if i[0]==emp_name:
50                 i[4]=depart_name
51
52     # Remove a department from the employee
53     def remove_department(self, emp_name, depart_name):
54         for i in self.List:
55             #check for user
56             if i[0]==emp_name:
57                 i[4]='NULL'
58 #setter and getter format
59     '''@property
60     def name(self):
61         return self.name
62
63     @property
64     def ID(self):
65         return self.EmpID
66     @property
67     def department(self):
68         return self.department
69     @property
70     def Designation(self):
71         return self.designation
72     @property
73     def experience(self):
74         return self.experience
75
76     @ID.setter
77     def ID(self, ID):
78         self.EmpID = ID
79
80     @department.setter
81     def department(self, department):
82         self.department = department
83     @name.setter
84     def name(self, name):
85         self.name = name
86
87     @Designation.setter
88     def Designation(self, designation):
89         self.Designation = designation
90
91     @experience.setter
92     def experience(self, experience):
93         self.experience = experience

```



```

94     '''
95 #Manager class
96 class Manager():
97     M=[]
98     #main list of manager
99     #initial class
100     def __init__(self):
101         pass
102
103     # Add manager to list
104     def addManager(self,name,position):
105         #CHECK if project manager
106         if position == 'Project_Manager':
107             department='NULL'
108             #create a dummy list
109             self.m=[name, position ,department]
110             #add to main list
111             self.M+=self.m
112             #if delivery manager more than 1 department can be added
113         elif position == 'Delivery_Manager':
114             department=[]
115             self.m=[name, position ,department]
116             self.M+=self.m
117             print("Manager Added")
118         # show manager
119     def showManager(self ,manager_name):
120         for i in self.M:
121             #check manager name
122             if i[0] ==manager_name:
123                 print(i[0] , i[1] , i[2])
124                 print("Manager Shown")
125                 break
126             print("Manager not present")
127         # find the manager name true if found
128     def find(self ,manager_name):
129         for i in self.M:
130             #check manager name
131             if (i[0]==manager_name):
132                 return True
133         return False
134
135     #check for existence of manager name
136     def exist(self ,manager_name):
137         for i in self.M:
138             if i[0]==manager_name:
139                 return True
140         return False
141     #check for existence of manager name
142     def exist_m(self ,emp_name):
143         for i in self.M:

```

```

144         if i[0]==emp_name:
145             #if project manager or not
146             if i[1]=="Project_Manager" and i[2]=='NULL':
147                 return True
148             else:
149                 print("Already Assignment")
150             #if delivery manager or not
151             if i[1]=="Delivery_Manager" :
152                 return True
153             return False
154         return False
155     #setter and getter commented
156     '''
157     @set_position.setter
158     def set_position(self,position):
159         self.position = position
160
161     @set_experience.setter
162     def set_experience(self,experience):
163         self.experience = experience
164     @addEmployee.setter
165     def addEmployee(self,name,ID,designation,department,experience):
166         super(Employee, self).__init__(name)
167         self.EmpID = ID
168         self.Designation = designation
169         self.department = department
170         self.experience =self.experience
171
172     @property
173     def get_ID(self):
174         return self.EmpID
175     @property
176     def get_department(self):
177         return self.department
178     @property
179     def get_Designation(self):
180         return self.designation
181     @property
182     def get_experience(self):
183         return self.experience
184     '''
185     #department class
186     class Department:
187         D=[]
188         #department list
189         def __init__(self):
190             pass
191
192     # add a department
193     def addDepartment(self,name,maxEmployee,manager):

```

```

194         empJoined=[]
195         #if manager exist
196         self.d=[name,maxEmployee,manager,empJoined]
197         self.D+=self.d]
198
199         #maxchange change the max change
200     def max_change(self,depart_name,max):
201         for i in self.D:
202             if i[0]==depart_name:
203                 i[1]=str(max)
204                 break
205
206     #show department
207     def showDepartment(self,depart_name):
208         for i in self.D:
209             if i[0]==depart_name:
210                 #check for the department
211                 print(i,end=" ")
212                 print("\nDepartment Shown")
213         print("Department not present")
214
215     ## existence of the department
216     def exist(self,depart_name):
217         for i in self.D:
218             if i[0]==depart_name:
219                 return True
220         return False
221
222     #max space check
223     def space(self,depart_name):
224         for i in self.D:
225             if i[0]==depart_name:
226                 count=len(i[3])
227                 if count < int(i[1]):
228                     return True
229                 else:
230                     return False
231
232     #add employ to department
233     def add_emp(self,depart_name,empl_name):
234         for i in self.D:
235             # check for department
236             if i[0]==depart_name:
237                 i[3].append(empl_name)
238                 # print(type(i[3]))
239                 break
240
241     # change the department
242     def change(self,depart_name,empl_name):
243         for i in self.D:

```

```

244         if i[0]==depart_name:
245             i[2]=empl_name
246             break
247
248     #remove the employee
249     def rem_emp(self ,depart_name ,empl_name):
250         for i in self.D:
251             if i[0]==depart_name:
252                 for emp in i[3]:
253                     if emp==empl_name:
254                         i[3].remove(empl_name)
255                         return True
256         print("Employee not Present")
257         return False
258
259     #not empty
260
261     def not_empty(self ,depart_name):
262         for i in self.D:
263             if i[0]==depart_name:
264                 count=len(i[3])#employ count
265                 if count != 0 :
266                     return True
267                 else:
268                     return False
269
270 filename = input("Enter the file name along with the path\n")
271 file = open(filename , "r")
272 #object of employee ,manager ,department
273 obj_emp = Employee()
274 obj_manag = Manager()
275 obj_depart = Department()
276 # get line from the file and parse the line
277 for line in file:
278     list=line[:-1].split(" ")
279     print(list)
280     #Add employee
281     if list[0] == 'ADDE':
282         #adding employee
283         print("ADDE\n")
284         #add employee
285         obj_emp.addEmployee(list[1] , list[2] , list[3] , list[4])
286     #Add manager
287     elif list[0] == 'ADDM':
288         #adding manager
289         print("ADDM\n")
290         #Add manager
291         obj_manag.addManager(list[1] , list[2])
292     # add Department
293     elif list[0] == 'ADDD':

```

```

294     flag=False
295     #find the managername
296     if obj_manag.find(list[3])==True:
297         flag=True
298     else:
299         flag=False
300     if flag== True:
301         # add Department
302         obj_depart.addDepartment(list[1], list[2], list[3])
303         print("Department added")
304     else:
305         print("Manager not found")
306         print("Department not added")
307     print("ADDD\n")
308
309     #Join the department
310 elif list[0] == 'JOIN':
311     #existence of department
312     if obj_depart.exist(list[1])==True:
313
314         if obj_depart.space(list[1]) ==True:
315             print("space present")
316             print("Person Added")
317             if obj_emp.exist(list[2]):
318                 print("Employee present")
319                 #add employ
320                 #add the department
321                 obj_depart.add_emp(list[1], list[2])
322                 obj_emp.add_department(list[2], list[1]) #2 =person
323             name l= depart name
324             else:
325                 print("employee not present")
326                 print("Person Not Added")
327             else:
328                 print("space not present")
329                 print("Person Not Added")
330             else:
331                 print("Person Not Added")
332
333             print("JOIN\n")
334             #leave the deprtment
335 elif list[0] == 'LEAVE':
336     #department exist
337
338     if obj_depart.exist(list[1])==True:
339         #not empty
340         if obj_depart.not_empty(list[1]) ==True:
341             print("space not empty")
342             if obj_emp.exist(list[2]):
343                 print("Employee present")

```

```

343         #remove employ
344         if obj_depart.rem_emp(list[1], list[2]) == True:
345             #update department
346             obj_emp.remove_department(list[2], list[1]) #2 =
person name 1= depart name
347             print("Employee removed")
348         else:
349             print("Employee not removed")
350     else:
351         print("employee not present")
352         print("Person Not Removed")
353     else:
354         print("space not present")
355         print("Person Not Removed")
356     else :
357         print("Person Not Removed")
358         print("LEAVE\n")
359         #modify the manger / of department
360     elif list[0] == 'MODIFY':
361         if obj_depart.exist(list[1]) == True:
362             #check if manger is not assigned any department or is
delivery manager
363             if obj_manag.exist_m(list[3]) == True:
364                 obj_depart.change(list[1], list[3])
365                 print("Manager Added")
366                 obj_depart.max_change(list[1], list[2])
367             else:
368                 obj_depart.max_change(list[1], list[2])
369                 print("Manager doesnot exist")
370         else:
371             print("Department Doesnot exist")
372             print("MODIFY\n")
373     #show employee
374     elif list[0] == 'SHOWE':
375         #show the employee if exist
376         obj_emp.showEmployee(list[1])
377         print("SHOWE\n")
378     elif list[0] == 'SHOWM':
379         #show the manager if exist
380         obj_manag.showManager(list[1])
381         print("SHOWM\n")
382     elif list[0] == 'SHOWD':
383         #show the department if exist
384         obj_depart.showDepartment(list[1])
385         print("SHOWD\n")
386     else :
387         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/>