File management

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

def ComputeSatisfaction(A_i,v_i):
    S=100
    for i in range(5):
        if A_i[i]==v_i:
            S=i

    return S
def ComputeOverallSatisfaction(A,v):

    Overall_Satisfaction_Score=0.0
    for i in range(A.shape[0]):
        Overall_Satisfaction_Score+=ComputeSatisfaction(A[i,:],v[i])

    return Overall_Satisfaction_Score

def perturb_assignment(v,non_assigned_projects):
    v_perturb=v.copy()
    non_assigned_projects_perturb=non_assigned_projects.copy()

    uni_0_1=np.random.uniform()

    if uni_0_1<0.5:
        shuffle_method=1
    else:
        shuffle_method=2


    if shuffle_method==1:
```

```
        student_i=np.random.randint(0,len(v))
        student_j=np.random.randint(0,len(v))
        v_perturb[student_i]=v[student_j]
        v_perturb[student_j]=v[student_i]
    else:
        # Randomly select index for student and non assigned project
        student_index=np.random.randint(0,len(v))
        project_index=np.random.randint(0,len(non_assigned_projects))

        # Swap student i project with one from unassigned list


        project_from_non_assigned=non_assigned_projects_perturb[project_index].copy()
        existing_project_st_i=v_perturb[student_index].copy()

        v_perturb[student_index]=project_from_non_assigned
        # Pass student is project onto unassigned list
        non_assigned_projects_perturb[project_index]=existing_project_st_i

        #print(project_index)
        #print(student_index)
        #print(non_assigned_projects[project_index])
        #print(v[student_index])


    #if (len(np.unique(v_perturb))!=len(v_perturb)):
    #     print(v_perturb)
    #     print(v)
    #if (len(np.unique(non_assigned_projects_perturb))!=len(non_assigned_projects_perturb)):
    #    print(v)
    #    print(np.sort(non_assigned_projects))
    #    print(v_perturb)
    #    print(np.sort(non_assigned_projects_perturb))


    #assert(len(np.unique(v_perturb))==len(v_perturb))
    #assert(len(np.unique(non_assigned_projects_perturb))==len(non_assigned_projects_perturb

    #print('Perturbed:')
    #print(v_perturb)
    #print('Orig:')
    #print(v)
```

```
    return v_perturb,non_assigned_projects_perturb

def Score_Assignment(A,v):

    preference_hit=np.zeros((1,A.shape[1]),dtype=int)
    print(preference_hit.shape)
    for i in range(A.shape[0]):
        for j in range(A.shape[1]):
            if A[i,j]==v[i]:
                preference_hit[0,j]+=1

    #Supervisor workload


    return np.squeeze(preference_hit)

def ComputeSupervisorWorkload(df_project,v):
    # Loop over students

    df_workload=pd.DataFrame(columns=['Supervisor','Num_projects'])

    unique_supervisors=df_project['Supervisor'].unique()

    df_workload['Supervisor']=pd.Series(unique_supervisors)
    df_workload['Num_projects']=0




    for i in range(len(v)):
        project_i=v[i]

    #    # Identify assigned project
        supervisor=(df_project['Supervisor'][df_project['ID']==project_i])
        supervisor=(np.squeeze(supervisor.to_numpy()))
        num_existing_projects=df_workload['Num_projects'][df_workload['Supervisor']==supervis


        df_workload['Num_projects'][df_workload['Supervisor']==supervisor]+=1
```

```
    #asdf
    return df_workload

    # Find supervisor  for project

    # Increment supervisor project count




use_data=2
# Load preference matrix
if use_data==1:
    A=np.array([[1, 4, 6, 5, 1],[1, 4, 5, 2, 3],[2, 5, 6, 7, 22]])
    num_students=A.shape[0]

    assigned_projects=np.array([1,4,2])
    non_assigned_projects=np.array([3,5,7,9,10,11,12,13])
else:
    str='../StudentPreferencesAY20242025-2.csv'
    df=pd.read_csv(str)
    A=df.iloc[:, 3:10]
    A=A.to_numpy()
    num_students=A.shape[0]

    FirstName=df['FirstName']
    SurName=df['Surname']

    project_str='../projects.csv'
    df_project=pd.read_csv(project_str)
    project_inds=df_project.iloc[:, 0]
    SupervisorName=df_project['Supervisor']



    project_inds=project_inds.to_numpy()
    assert(len(np.unique(project_inds))==len(project_inds))
    assigned_projects=np.random.choice(project_inds,num_students,replace=False)
    non_assigned_projects=np.setdiff1d(project_inds,assigned_projects)
    #assert(len(np.intersect(non_assigned_projects,assigned_projects))==0)
```

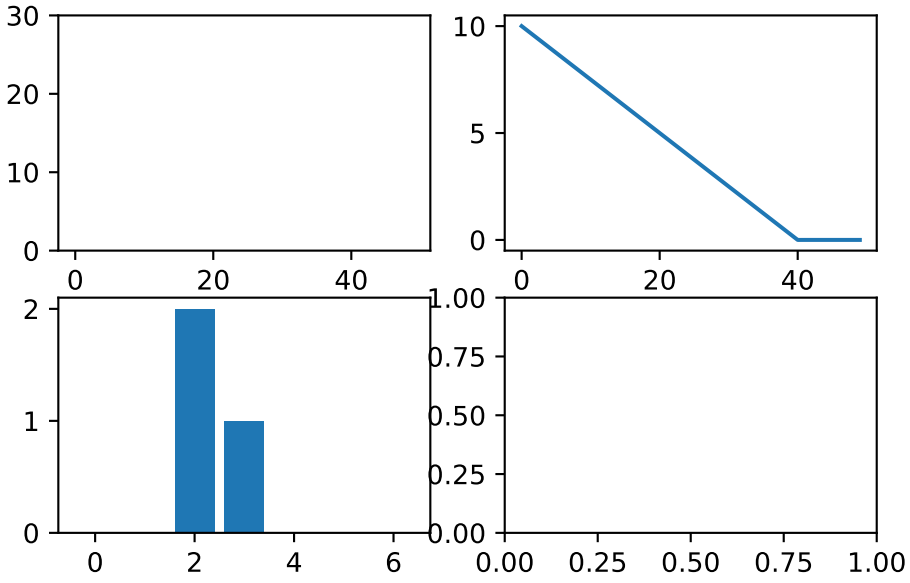
```
i=2
Satisfaction_Score=ComputeSatisfaction(A[2],assigned_projects[2])

overall_satisfaction_Score=ComputeOverallSatisfaction(A,assigned_projects)


# S
max_Temp=10.0
num_its=50
it_vec=np.arange(num_its)
Satisfation_score=np.zeros((num_its,1))
Temp_profile=np.zeros_like(Satisfation_score)
Temp_profile=max_Temp*(1-it_vec/(0.8*num_its))
Temp_profile[Temp_profile<0.0]=0.001
for it_ind in it_vec:
    print(it_ind)

    o_sat_sc=ComputeOverallSatisfaction(A,assigned_projects)

    # Perturb v

    v_perturb,non_assigned_projects_perturb=perturb_assignment(assigned_projects,non_assigned

    # Recompute overall_satisfaction

    perturbed_o_sat_sc=ComputeOverallSatisfaction(A,v_perturb)

    do_print=False
    if do_print==True:
        print('Assignments')
        print(assigned_projects)
        print(v_perturb)
        print('Scores:')
        print(o_sat_sc)
        print(perturbed_o_sat_sc)


    # Accept with some prob

    if perturbed_o_sat_sc<o_sat_sc:
        assigned_projects=v_perturb
```

```
        o_sat_sc=perturbed_o_sat_sc
        non_assigned_projects=non_assigned_projects_perturb
    else:
        Delta_S=perturbed_o_sat_sc-o_sat_sc
        Delta_S=Delta_S
        T=Temp_profile[it_ind].astype(float)
        prob_acc=np.exp(-1.0*Delta_S/T)
        r=np.random.uniform()
        if r<prob_acc:
            assigned_projects=v_perturb
            o_sat_sc=perturbed_o_sat_sc
            non_assigned_projects=non_assigned_projects_perturb



    # Record satisfaction score
    Satisfation_score[it_ind]=o_sat_sc

assigned_project_counts=Score_Assignment(A,assigned_projects)
df_num_projects=ComputeSupervisorWorkload(df_project,assigned_projects)

fig,ax=plt.subplots(2,2)
ax[0,0].plot(Satisfation_score)
ax[0,0].set_ylim([0, 30])
ax[0,1].plot(Temp_profile)

project_ind=np.arange(A.shape[1])
ax[1,0].bar(project_ind,assigned_project_counts)

#print(assigned_projects)
assert(len(np.unique(assigned_projects))==len(assigned_projects))

df_assigned_projects=pd.DataFrame(columns=['FirstName','SurName','AssignedProject','Supervis
df_assigned_projects['FirstName']=FirstName
df_assigned_projects['SurName']=SurName
df_assigned_projects['AssignedProject']=assigned_projects


df_assigned_projects = pd.merge(left=df_assigned_projects, right=df_project, left_on='Assigne
```

```
/var/folders/m_/vc0kz_0x6ls5n4qnksq052jw0000gp/T/ipykernel_91519/822496710.py:210: RuntimeWa
```

```
invalid value encountered in scalar divide

/var/folders/m_/vc0kz_0x6ls5n4qnksq052jw0000gp/T/ipykernel_91519/822496710.py:117: SettingWi


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

```
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
(1, 7)
```



The original preference matrix was:

	FirstName	Surname	Pref 1	Pref 2	Pref 3
0	Panayiotis	Gerolemou	34	40	42
1	Calum	Jackson	58	8	39
2	Katheryn	Todd	9	58	1
3	Bhagat	Athwal	42	4	45
4	Emily	Laycock	40	1	133
5	Stephen	Mainds	122	22	62
6	Abigail	Mir	134	125	16
7	Callum	Robertson	4	150	8
8	Megan	McGlone	34	41	8
9	Saif	Akhtar	4	45	42
10	Joseph	Stasaitis	133	45	58
11	Emily	Rice	26	46	62
12	Lucy	Wright	1	40	22
13	Andrew	Nimmo	4	39	40
14	Gregor	Downs	8	1	14

The project assignment to students is:

	FirstName	SurName	AssignedProject	Supervisor_x	ID
0	Panayiotis	Gerolemou	18	NaN	18
1	Calum	Jackson	39	NaN	39
2	Katheryn	Todd	52	NaN	52
3	Bhagat	Athwal	43	NaN	43
4	Emily	Laycock	36	NaN	36
5	Stephen	Mainds	21	NaN	21
6	Abigail	Mir	34	NaN	34
7	Callum	Robertson	45	NaN	45
8	Megan	McGlone	29	NaN	29
9	Saif	Akhtar	60	NaN	60
10	Joseph	Stasaitis	11	NaN	11
11	Emily	Rice	13	NaN	13
12	Lucy	Wright	22	NaN	22
13	Andrew	Nimmo	31	NaN	31
14	Gregor	Downs	124	NaN	124

The distribution of projects amongst staff is:

	Supervisor	Num_projects
0	NaN	0
1	Agis Athanassoulis (Staff)	0
2	Dumitru Trucu (Staff)	3
3	Eric Hall (Staff)	1
4	Ewa Bieniecka (Staff)	1
5	Gunnar Hornig (Staff)	1
6	Hiroko Kamei (Staff)	0
7	Jeremy Parker (Staff)	0
8	John McDermott (Staff)	3
9	Karen Meyer (Staff)	1
10	Niall Dodds (Staff)	1
11	Philip Murray (Staff)	1
12	Ping Lin (Staff)	2
13	Scott Gregory (Staff)	0