ELG5142[EG] - Ubiquitous Sensing / Smart Cities

Assignment 1

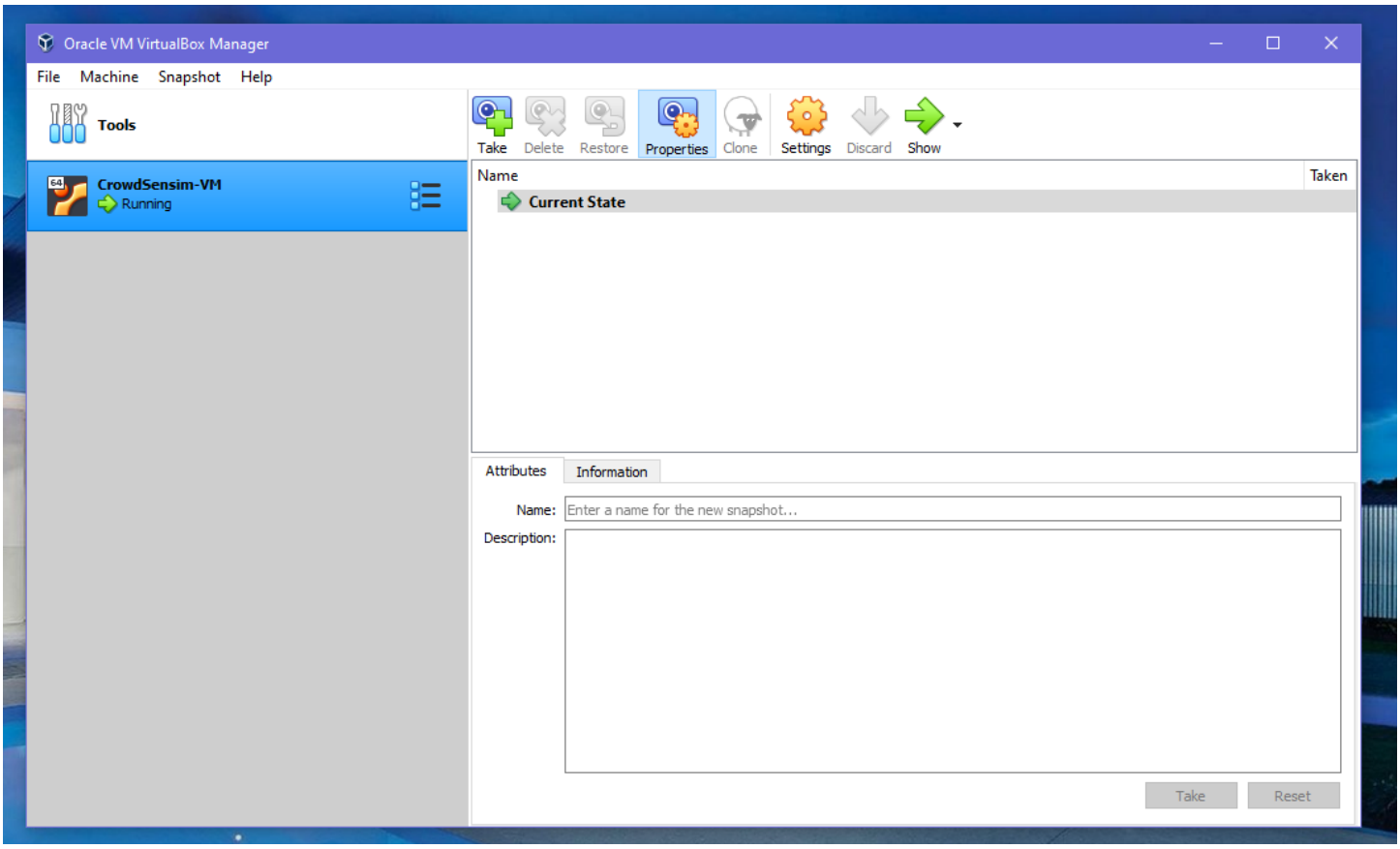
Group 23: Assignment 1

Assignment Steps that we are followed:

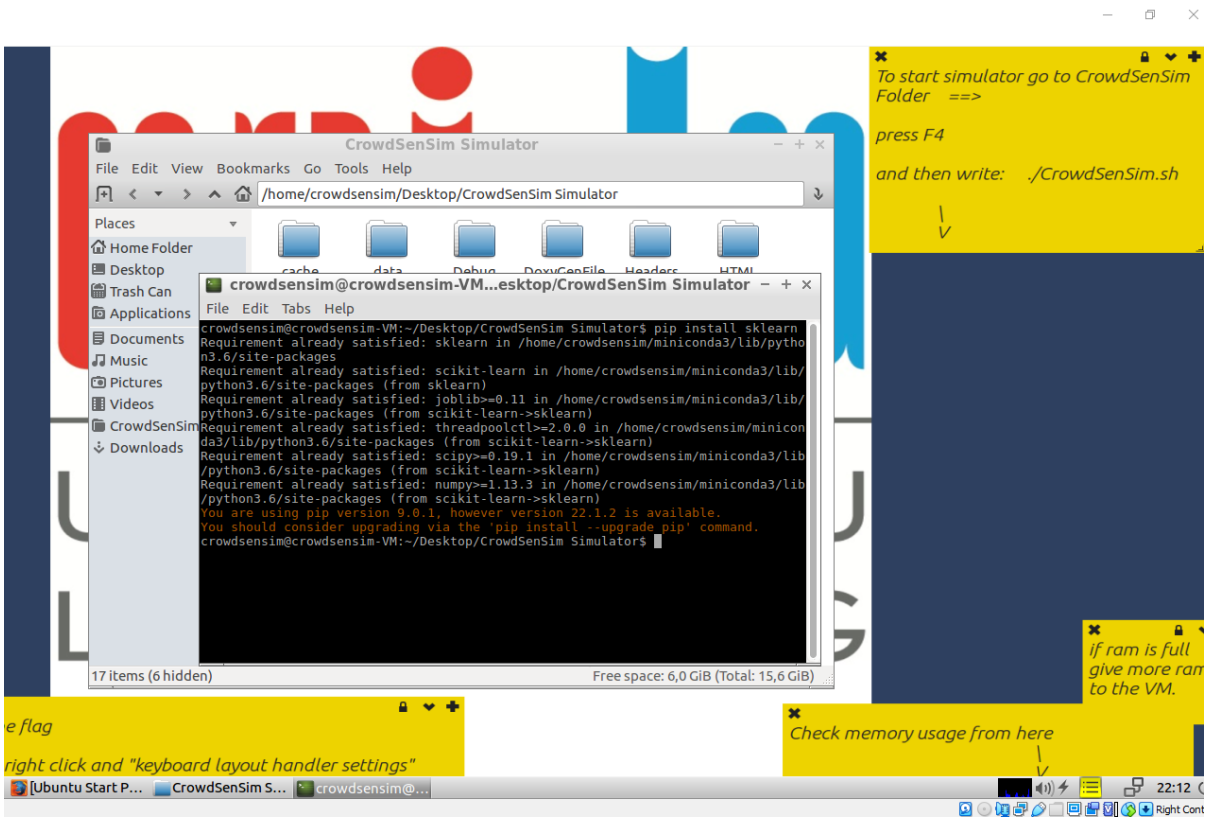
1. **Setup test environment by installing python and related libraries, install virtual machine and load CrowdSenSim image.**

1) Generating Tasks.

Frist, we downloaded VirtualBox and import CrowdSenSim image.

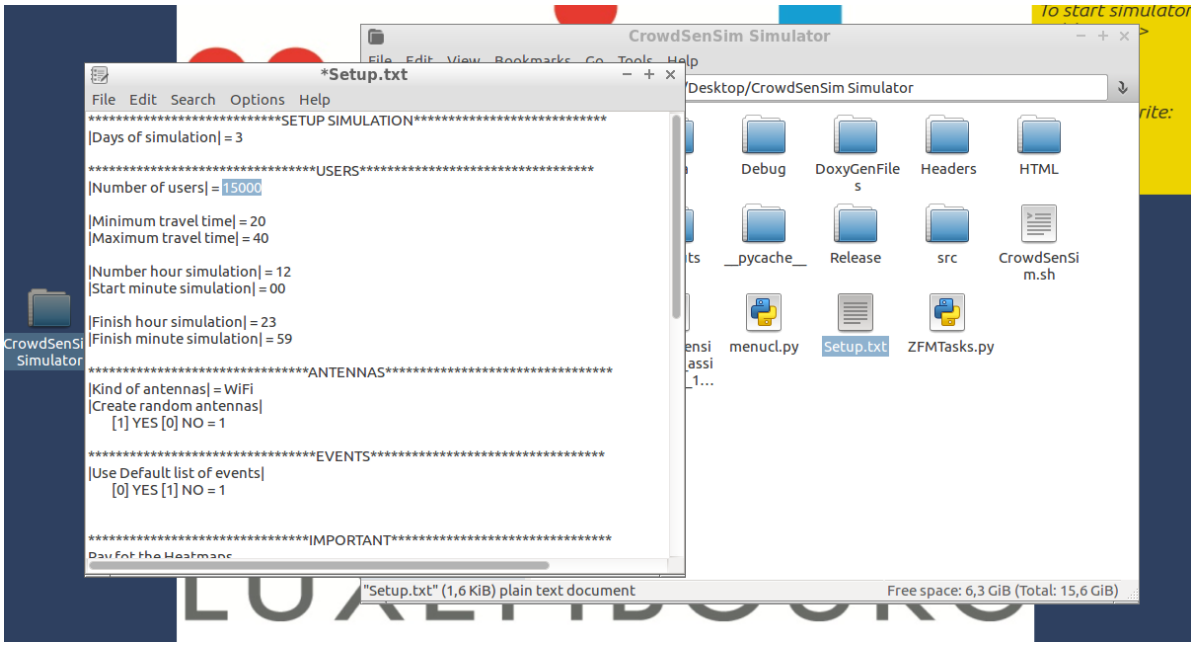


After that we downloaded some python libraries like sklearn.



2. **Configure Setup.txt file accordingly. The number of days and the number of users is configured in Setup.txt file.**

After that we have configured the setup file based on the assignment requirements.



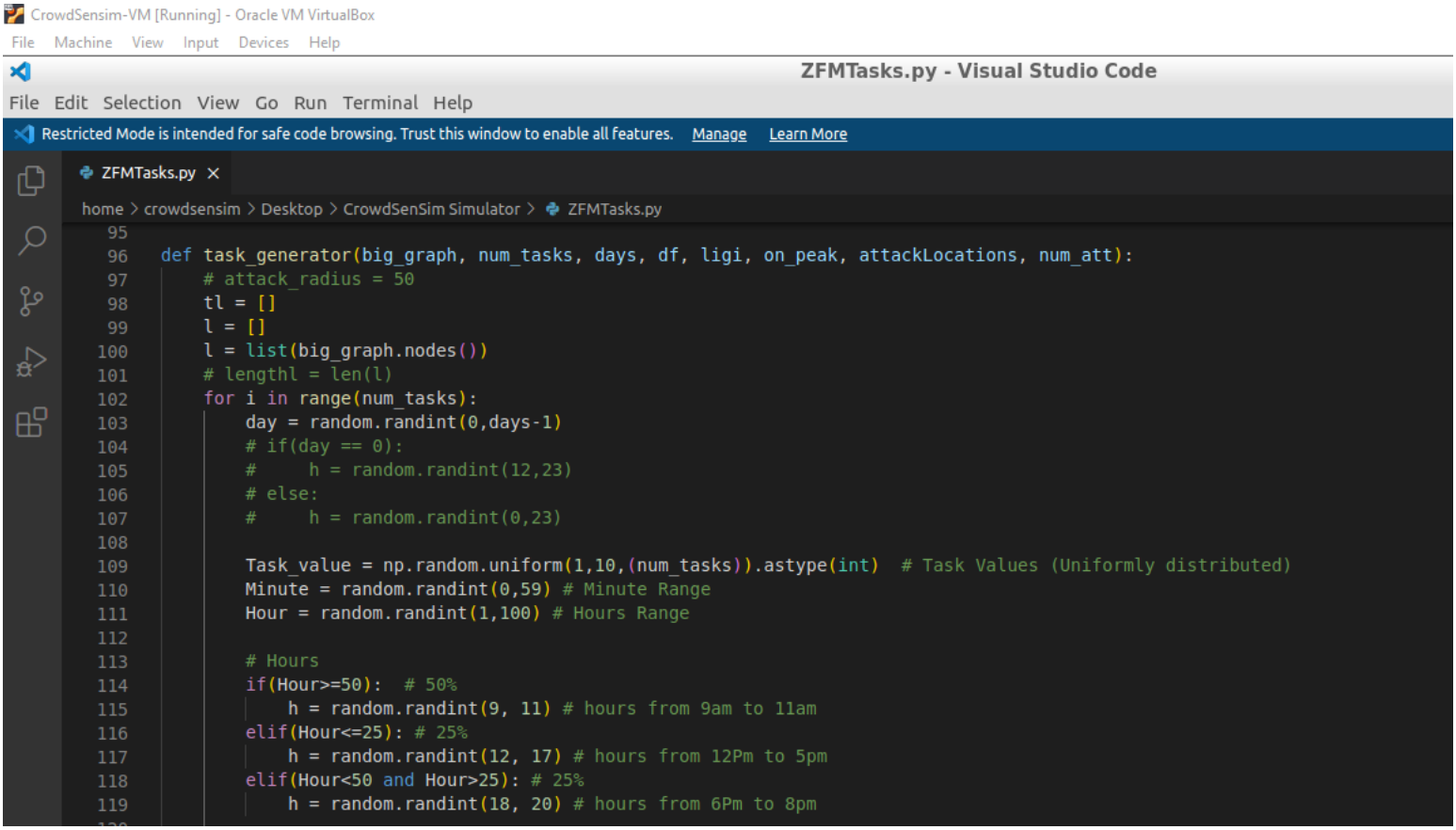
3. Refer to ZFMTasks.py to generate tasks and save in mytask.txt

And then we have changed the task_generator function that in ZFMTasks.py based on the assignment requirenment.

Table 1 Task generation requirements

Task Feature	Requirement
Day	Distribution consistent in [1, 2, 3]
Hour	50%: 9:00 AM-11:00 AM 25%: 12:00 PM-5:00 PM 25%: 6:00 PM-8:00 AM
Duration (minutes)	50% in {20, 40, 60} 30% in {30, 50, 70} 20% in {10, 80, 100}
Task Value	Uniformly distributed in [1,10]

task_generator function after the modifications that we have made on it.



CrowdSensim-VM [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

ZFMTasks.py - Visual Studio Code

File Edit Selection View Go Run Terminal Help

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

ZFMTasks.py

home > crowdsensim > Desktop > CrowdSenSim Simulator > ZFMTasks.py

```
116             elif(Hour<=25): # 25%
117                 h = random.randint(12, 17) # hours from 12Pm to 5pm
118             elif(Hour<50 and Hour>25): # 25%
119                 h = random.randint(18, 20) # hours from 6Pm to 8pm
120
121             # Duration
122             dur = random.randint(1,10)
123             if(dur<6): # 50%
124                 duration = random.choice([20, 40, 60]) # {20, 40, 60}
125             elif(dur<9 and dur<=6): # 30%
126                 duration = random.choice([30, 50, 70]) # {30, 50, 70}
127             else: # 20%
128                 duration = random.choice([10, 80, 100]) # {10, 80, 100}
129
130
131             remaining_t = duration
132             resources = random.randint(7,10)
133             index = random.choice(range(len(l)))
134             y=big_graph.node[l[index]]['y']
135             x=big_graph.node[l[index]]['x']
136             grid_num=convert_location(big_graph,y,x)
137
138             tl.append([i+1, float(y), float(x), day, h, Minute, duration, remaining_t, resources, df,ligi,on_peak,grid_num,Task_value[i]])
139
140         return tl
```

Mytask.txt and task_value is uniformly distributed from 1 to 10

mytask.txt - Notepad

File Edit Format View Help

ID-Task/	-/Lat/	-/Long/	-/Day/	-/Hour/	-/Minute/	-/Duration/	-/Remaining time/	-/Resources/	-/Coverage/	-/Ligitimacy/	-/on peak hour/	-/grid_number/	-/task_value
1	42.12226496514614	12.534513865571489	2	14	51	60	60	8	100	True	True	19	3
2	42.15576890373553	12.495746059653097	2	11	40	80	80	10	100	True	True	65	4
3	42.14397482701999	12.506639458908973	1	10	3	80	80	8	100	True	True	46	6
4	42.14299031339726	12.49604406184683	2	10	39	60	60	9	100	True	True	45	1
5	42.14777241495238	12.521271239709776	2	10	43	100	100	8	100	True	True	58	8
6	42.145508959947676	12.466538268302731	0	9	45	60	60	9	100	True	True	51	2
7	42.121752603039944	12.534938453003988	0	11	29	10	10	9	100	True	True	19	1
8	42.14396601555058	12.464758061491924	0	11	57	60	60	7	100	True	True	41	5
9	42.16056135650676	12.525319716537046	2	10	42	40	40	9	100	True	True	78	3
10	42.1521923370804	12.477919367932504	0	18	25	60	60	10	100	True	True	62	3
11	42.13087936365291	12.478966118155453	1	16	25	20	20	10	100	True	True	22	3
12	42.154980432236144	12.50515678581365	1	10	12	10	10	9	100	True	True	66	4
13	42.154644161678824	12.527088115633667	2	11	26	80	80	8	100	True	True	68	3
14	42.14911953906333	12.506205614861857	0	11	37	100	100	8	100	True	True	56	1
15	42.15069187675908	12.492931807132631	0	20	31	30	30	7	100	True	True	64	3
16	42.156454917090514	12.500553781397018	2	10	56	80	80	7	100	True	True	75	5
17	42.14412738934361	12.47867288243402	0	13	40	80	80	9	100	True	True	42	7
18	42.13629943227805	12.4930267810776	0	11	27	10	10	7	100	True	True	34	4
19	42.14841517291063	12.477475298289447	0	9	31	20	20	9	100	True	True	52	7
20	42.14670219474318	12.509569222881069	2	9	20	20	20	7	100	True	True	56	7
21	42.15669033445653	12.492250893757234	0	10	31	80	80	10	100	True	True	74	2
22	42.14930089262161	12.516585221285435	0	11	9	80	80	7	100	True	True	57	7
23	42.140585042594 58	12.527607108447398	2	19	8	40	40	10	100	True	True	49	7
24	42.14718200796527	12.492278901129529	0	10	59	40	40	8	100	True	True	54	1
25	42.14487185910861	12.478374718445254	1	9	47	20	20	9	100	True	True	52	4
26	42.135803698566576	12.48560290421483	0	13	14	40	40	9	100	True	True	33	7
27	42.15654320206294	12.485188079369696	0	9	11	20	20	7	100	True	True	73	3
28	42.13050834495915	12.521232399797368	1	19	25	80	80	10	100	True	True	28	2
29	42.13237601612404	12.516513976138306	0	11	41	80	80	9	100	True	True	27	6

Mytask.csv and task_value is uniformly distributed from 1 to 10

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ID	Latitude	Longitude	Day	Hour	Minute	Duration	RemainingTime	Resources	Coverage	Ligitimacy	OnPeakHours	GridNumber	task_value
2	1	42.12226497	12.53451	2	14	51	60	60	8	100	TRUE	TRUE	19	3
3	2	42.1557689	12.49575	2	11	40	80	80	10	100	TRUE	TRUE	65	4
4	3	42.14397483	12.50664	1	10	3	80	80	8	100	TRUE	TRUE	46	6
5	4	42.14299031	12.49604	2	10	39	60	60	9	100	TRUE	TRUE	45	1
6	5	42.14777241	12.52127	2	10	43	100	100	8	100	TRUE	TRUE	58	8
7	6	42.14550896	12.46654	0	9	45	60	60	9	100	TRUE	TRUE	51	2
8	7	42.1217526	12.53494	0	11	29	10	10	9	100	TRUE	TRUE	19	1
9	8	42.14396602	12.46476	0	11	57	60	60	7	100	TRUE	TRUE	41	5
10	9	42.16056136	12.52532	2	10	42	40	40	9	100	TRUE	TRUE	78	3
11	10	42.15219234	12.47792	0	18	25	60	60	10	100	TRUE	TRUE	62	3
12	11	42.13087936	12.47897	1	16	25	20	20	10	100	TRUE	TRUE	22	3
13	12	42.15498043	12.50516	1	10	12	10	10	9	100	TRUE	TRUE	66	4
14	13	42.15464416	12.52709	2	11	26	80	80	8	100	TRUE	TRUE	68	3
15	14	42.14911954	12.50621	0	11	37	100	100	8	100	TRUE	TRUE	56	1
16	15	42.15069188	12.49293	0	20	31	30	30	7	100	TRUE	TRUE	64	3
17	16	42.15645492	12.50055	2	10	56	80	80	7	100	TRUE	TRUE	75	5
18	17	42.14412739	12.47867	0	13	40	80	80	9	100	TRUE	TRUE	42	7
19	18	42.13629943	12.49303	0	11	27	10	10	7	100	TRUE	TRUE	34	4
20	19	42.14841517	12.47748	0	9	31	20	20	9	100	TRUE	TRUE	52	7
21	20	42.14670219	12.50957	2	9	20	20	20	7	100	TRUE	TRUE	56	7
22	21	42.15669033	12.49225	0	10	31	80	80	10	100	TRUE	TRUE	74	2
23	22	42.14930089	12.51659	0	11	9	80	80	7	100	TRUE	TRUE	57	7
24	23	42.14058504	12.52761	2	19	8	40	40	10	100	TRUE	TRUE	49	7
25	24	42.14718201	12.49228	0	10	59	40	40	8	100	TRUE	TRUE	54	1
26	25	42.14487186	12.47837	1	9	47	20	20	9	100	TRUE	TRUE	52	4
27	26	42.1358037	12.4856	0	13	14	40	40	9	100	TRUE	TRUE	33	7
28	27	42.1565432	12.48519	0	9	11	20	20	7	100	TRUE	TRUE	73	3
29	28	42.13050834	12.52123	1	19	25	80	80	10	100	TRUE	TRUE	28	2

Task_value

```
109    Task_value = np.random.uniform(1,10,(num_tasks)).astype(int)  # Task Values (Uniformly distributed)
```

2) Obtaining user movement event.

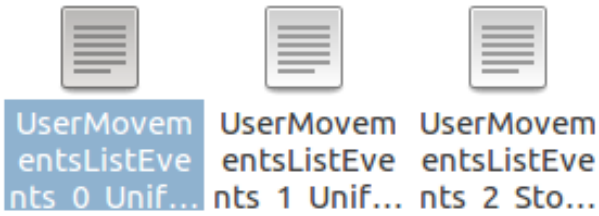
4. **Refer to crowdsensim2.py to generate user movements event and save in to usermovements_0.txt, usermovements_1.txt, and usermovements_2.txt (representing three-day simulation)**

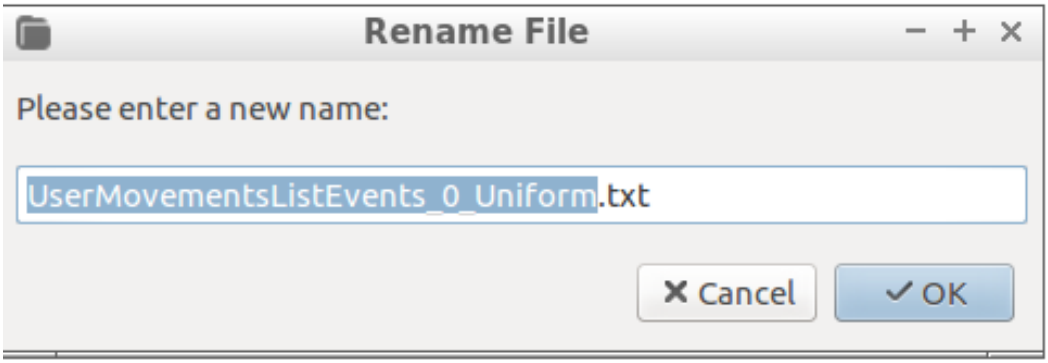
• Generate user mobility under uniform algorithm

- Uniform algorithm

```
626         speed = random.uniform(min_speed,max_speed)
627         mincamm=random.randint(20,40)
628         cut=mincamm*60*speed
629         cutadded=cut+maxlen
630
631         (length, path)= nx.single_source_dijkstra(G_old, origin_node,target=None, cutoff=cutadded, weight='length')
632
633             idr=0
634             for l in length:
635                 if length[l]>cut:
636                     idr=l
637
638             if idr == 0:
639                 idr = max(length, key=length.get)
640
641         route=path[idr]
642
643         #get the last node so we can continue our path!!
644
645         # If userused==1, do the HTML pathway generation for it
646         # most important part of this refactor, we can now do this for any path
647         if userused==1 :
648             htmlRouteGen(route, G_old, save=toSave)
```

The generated files using uniform algorithm

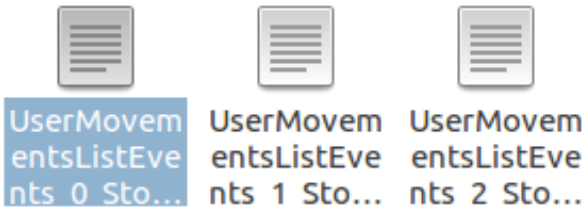


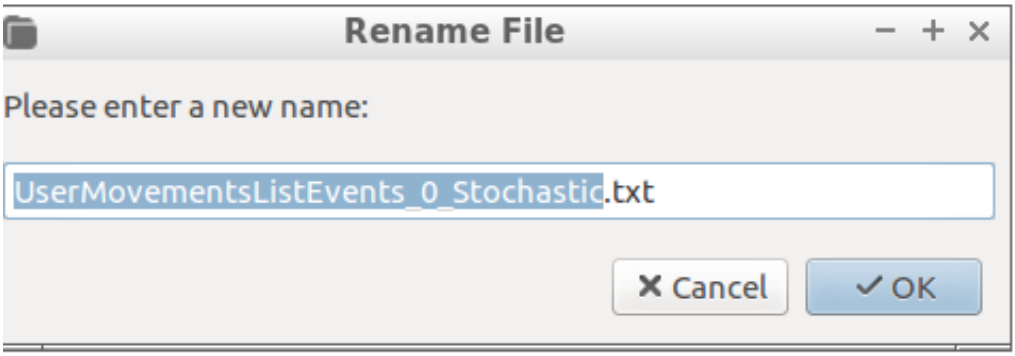


- **Generate user mobility under stochastic algorithm**
  - Stochastic algorithm

```
            speed = random.uniform(min_speed,max_speed)
            mincamm=random.randint(20,40)
            cut=mincamm*60*speed
            cutadded=cut+maxlen

            route = None
            if (not social_model_routes):
                try:
                    (length, path)= nx.single_source_dijkstra(G_old, origin_node,target=None, cutoff=cutadded, weight='length')
                except:
                    continue

                idr = 0
                if stochastic_model_routes:
                   Lengths_Array = np.array(list(length.values()))
                   Lengths_Array = np.where(Lengths_Array <= 0, 0.1, Lengths_Array)
                   Weights = np.random.dirichlet(Lengths_Array , 1)
                   idr = random.choices(tuple(length), weights = Weights[0])[0]

                else:
                    for l in length:
                        if length[l]>cut:
                            idr=l
                    if idr==0:
                        idr=max(length, key=length.get)

                route=path[idr]
            else:
                route = socialModel.chooseRoute(userused-1, cut, G_old)
```

The generated files using Stochastic algorithm





Results of Stochastic and uniform algorithms.

For example, this is the first file **(different file for each algorithm)** and it has 3000 users.