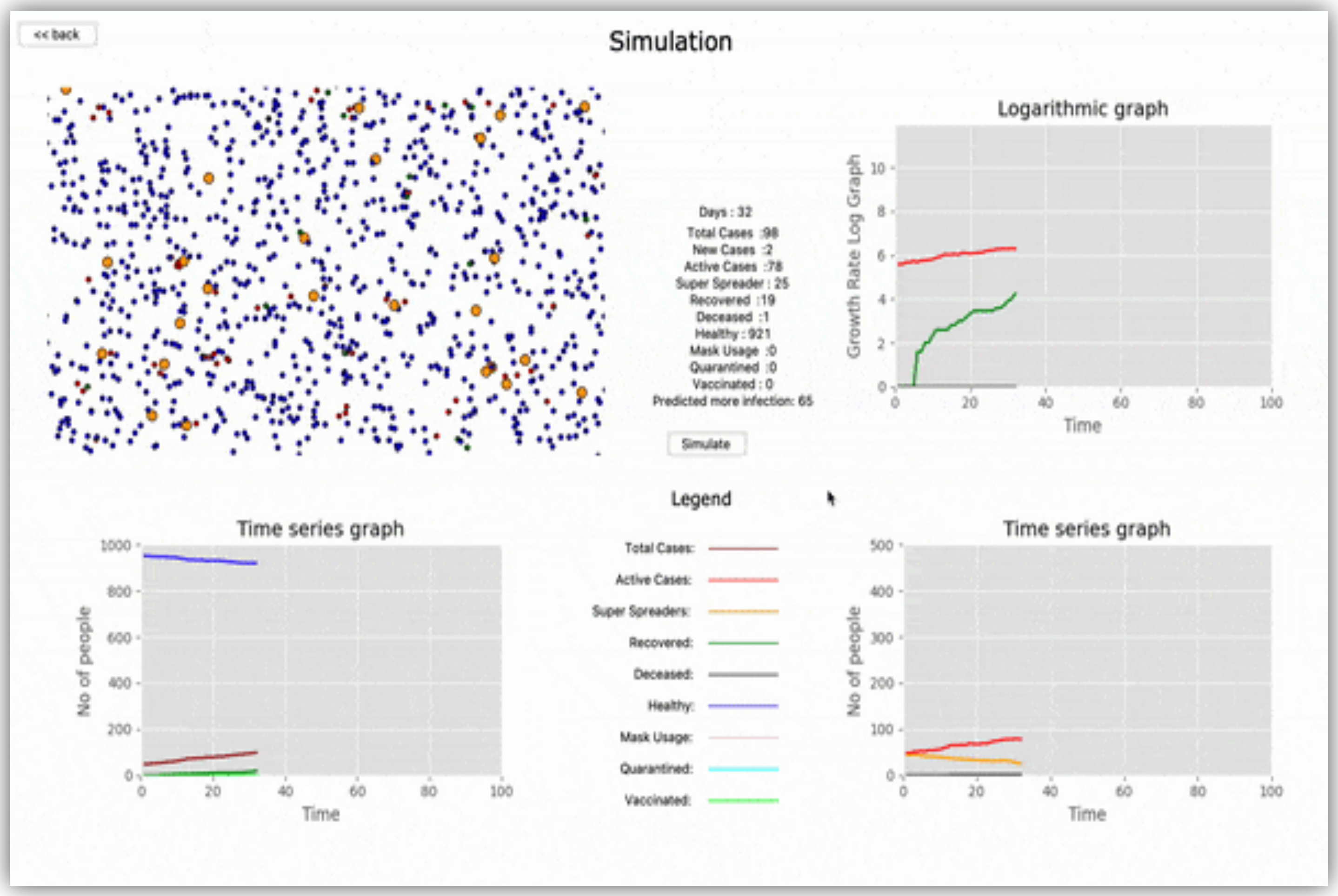
COVID-19 Simulation

Prepared for:

PSA Final Project – INFO 6205 Spring 2021



```
Days : 32
Total Cases :98
New Cases :2
Active Cases :78
Super Spreader : 25
Recovered :19
Deceased :1
Healthy : 921
Mask Usage :0
Quarantined :0
Vaccinated : 0
Predicted more infection: 65
```

Prepared by:

Siddhartha Raju (001084614)

Gautham Rajsimha Pulipati (001572432)

Naveen Kumar Buddhala (001582394)

Table of Contents

INTRODUCTION

Simulation model of a disease projects how infectious diseases progress i.e how its components like transmission, spread and control of any infection grows over time and how it may strongly or weakly effect the dynamics of an epidemic. It also helps to inform public health interventions.
Simulation model gives a pictoral or graphical way and it involves many factors, actors and components.
This is a new method of understanding and teaching about any disease outbreaks.
This can save money,lives and time by allowing us to test it without even applying it to real world.
Our simulation model fits for a particular pattern which is a general method involving R and K factor of the disease and several other factors of its spread.

AIM

Simulate the spread of a virus with respect to several factors and introducing various control measures later during the simulation and finally analysing the virus.
Simulations will take into account:

- The R factor and k factor of the disease
- The usage and effectiveness of masks
- The prevalence of testing and contact tracing
- The availability and efficacy of the vaccine
- Any barriers to entry (including quarantining) into the subject area.

Building a User Interface to set several configuration values involved for respective diseases and simulating over a population and finally analysing through various graphs.

PROJECT DETAILS

This disease simulation model is implemented in Python using TkInter and numpy library for probability association and matplotlib library for plotting the graph. The project consists of the following things

File Structure:

1. Config.cfg

a. The configuration file for the project can be found at the following location ~/config/config.cfg. The config file is used to set the default values for various diseases.
b. In case no values are provided through the interface, the application will self-feed the values from this configuration file.

2. Config.py

a. This is a Singleton class, and it contains all the configuration values required to simulate the disease
b. It has the following properties:
c. property-name
d. population
e. initial-infected
f. r-factor
g. k-factor
h. days-contagious
i. mask introduction timelines
j. mask effectiveness
k. mask usage percentage
l. quarantine introduction timelines
m. quarantine effectiveness
n. quarantine usage percentage
o. vaccine introduction timelines
p. vaccine effectiveness
q. vaccine usage percentage

3. ConfigUtil.py

a. This is a Singleton class, and it contains the methods to load config file and get config values based on section and key passed.

4. Person.py

This Class contains all the person related information.

a. It contains the following properties along with all their respective getters and setters:
b. id
c. age
d. gender
e. x-coordinate
f. y-coordinate
g. vaccinated
h. quarantined
i. mask-usage
j. infected
k. recovered
l. deceased
m. recovery-days
n. can-infect-others

5. PersonUtil.py

a. The class contains method to perform update operations of the Person object like updating location and status quo about mask, quarantine and infection. The class is used for updating various parameters of a person object during simulation.

6. DataUtil.py

a. The class contains methods to perform data related operations like counting infected people, recovered people.

7. SimulationData.py

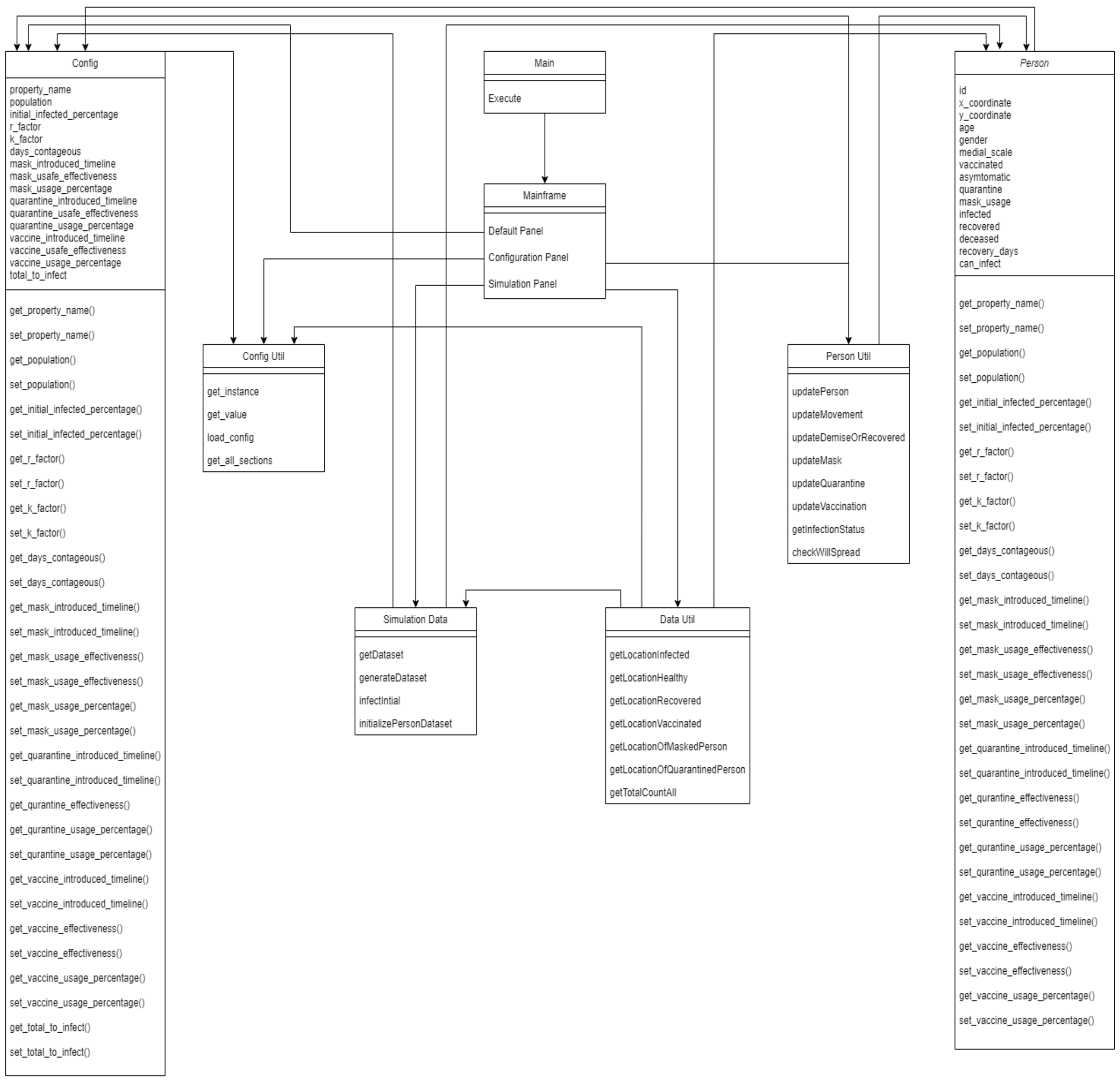
a. This class is used to setup the initial environment for simulation. It generates the initial dataset which is consumed during simulation for further processing.

8. Main.py

a. This is the main class which is executed at the program execution

9. MainFrame.py

a. This class consumes the data provided from Simulation Data and use it to visualize UI which contains Tkinter screens and graph for the user interface.


Config
property_name
population
initial_infected_percentage
r_factor
k_factor
days_contageous
mask_introduced_timeline
mask_usafe_effectiveness
mask_usage_percentage
quarantine_introduced_timeline
quarantine_usafe_effectiveness
quarantine_usage_percentage
vaccine_introduced_timeline
vaccine_usafe_effectiveness
vaccine_usage_percentage
total_to_infect
get_property_name()
set_property_name()
get_population()
set_population()
get_initial_infected_percentage()
set_initial_infected_percentage()
get_r_factor()
set_r_factor()
get_k_factor()
set_k_factor()
get_days_contageous()
set_days_contageous()
get_mask_introduced_timeline()
set_mask_introduced_timeline()
get_mask_usage_effectiveness()
set_mask_usage_effectiveness()
get_mask_usage_percentage()
set_mask_usage_percentage()
get_quarantine_introduced_timeline()
set_quarantine_introduced_timeline()
get_qurantine_effectiveness()
get_qurantine_usage_percentage()
set_qurantine_usage_percentage()
get_vaccine_introduced_timeline()
set_vaccine_introduced_timeline()
get_vaccine_effectiveness()
set_vaccine_effectiveness()
get_vaccine_usage_percentage()
set_vaccine_usage_percentage()
get_total_to_infect()
set_total_to_infect()
Main
Execute
Mainframe
Default Panel
Configuration Panel
Simulation Panel
Config Util
get_instance
get_value
load_config
get_all_sections
Person Util
updatePerson
updateMovement
updateDemiseOrRecovered
updateMask
updateQuarantine
updateVaccination
getInfectionStatus
checkWillSpread
Simulation Data
getDataset
generateDataset
infectIntial
initializePersonDataset
Data Util
getLocationInfected
getLocationHealthy
getLocationRecovered
getLocationVaccinated
getLocationOfMaskedPerson
getLocationOfQuarantinedPerson
getTotalCountAll
Person
id
x_coordinate
y_coordinate
age
gender
medial_scale
vaccinated
asymtomatic
quarantine
mask_usage
infected
recovered
deceased
recovery_days
can_infect
get_property_name()
set_property_name()
get_population()
set_population()
get_initial_infected_percentage()
set_initial_infected_percentage()
get_r_factor()
set_r_factor()
get_k_factor()
set_k_factor()
get_days_contageous()
set_days_contageous()
get_mask_introduced_timeline()
set_mask_introduced_timeline()
get_mask_usage_effectiveness()
set_mask_usage_effectiveness()
get_mask_usage_percentage()
set_mask_usage_percentage()
get_quarantine_introduced_timeline()
set_quarantine_introduced_timeline()
get_qurantine_effectiveness()
set_qurantine_effectiveness()
get_qurantine_usage_percentage()
set_qurantine_usage_percentage()
get_vaccine_introduced_timeline()
set_vaccine_introduced_timeline()
get_vaccine_effectiveness()
set_vaccine_effectiveness()
get_vaccine_usage_percentage()
set_vaccine_usage_percentage()

1 - Class diagram

Factors:

The following factors have been considered while designing the disease simulation

1. K factor

a. The dispersion parameter, it tells exactly how many are infected by each person i.e. in case of COVID-19 10-15 % of the population is responsible for 80% of the infections.

b. These people are called super-spreaders. Whereas others, may be quarantined and infect none.

2. Mask

a. 3 factors for mask are considered: introduction timeline, effectiveness, and percentage usage

b. Introduction timeline specifies number of days after which the masks were introduced i.e. after certain time of start of simulation

c. Effectiveness specifies how much does mask help us to protect from getting infected on a scale of 100, 0 signifies mask has no impact on in the virus to be infected

d. Percentage Usage specifies how many from the current population are using the masks

3. Quarantine

a. 3 factors for quarantine are considered: introduction timeline, effectiveness, and percentage usage

b. Introduction timeline specifies number of days after which the people started to get quarantined i.e. after certain time of start of simulation

c. Effectiveness specifies how much does quarantine help us to protect from getting infected on a scale of 100, 100 signifies if a person is quarantined, he cannot get infected

d. Percentage Usage specifies how many from the current population are quarantining after getting infected

4. Vaccine

a. 3 factors for vaccine are considered: introduction timeline, effectiveness, and percentage usage

b. Introduction timeline specifies number of days after which the people started to get vaccinated i.e. after certain time of start of simulation

c. Effectiveness specifies how much does vaccination help us to protect from getting infected on a scale of 100, 100 signifies if a person is vaccinated, he cannot get infected

d. Percentage Usage specifies how many from the current population are vaccinated

5. Days contagious

a. No. of days the person is infected, after catching the infection

6. Initial infected percentage

a. Percentage of total population to be infected at the beginning of simulation

IMPLEMENTATION

Factors considered:

R factor

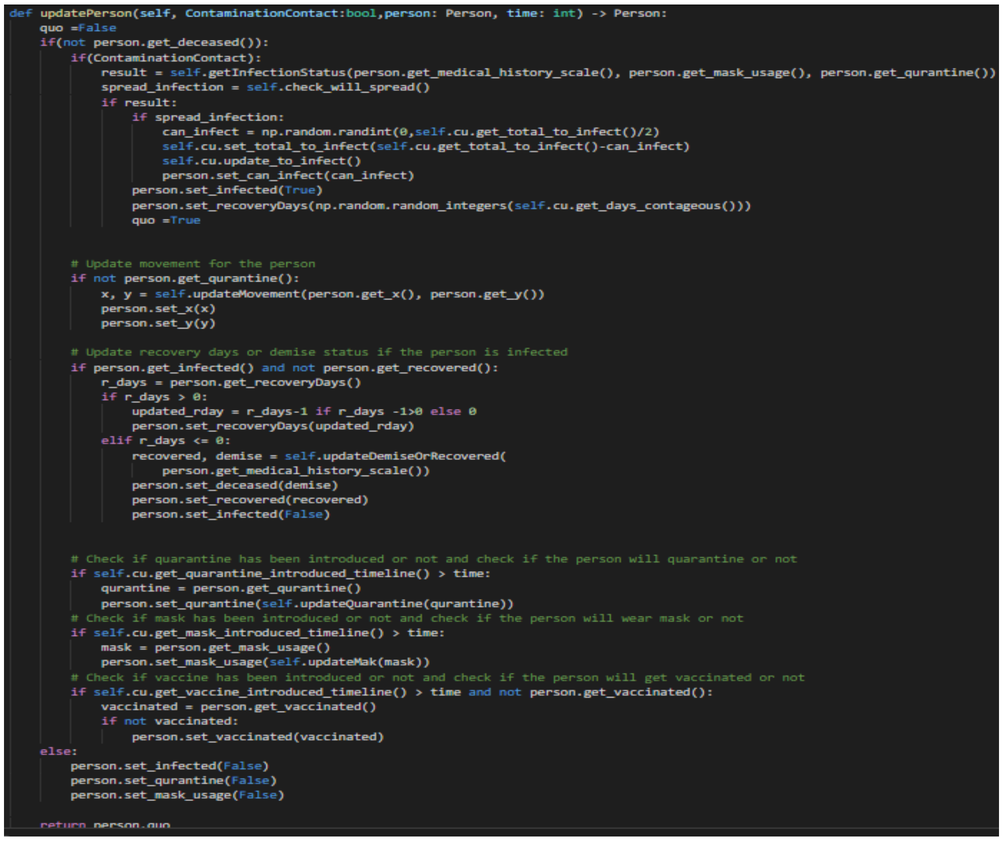
The average number of infections by one infected person.

GUI

There is a main panel which has 2 buttons: Set configuration and Start Simulation

These 2 buttons when clicked opens 2 other panels, one has all the default set configuration details which can be reset as per the user, while the other starts the simulation process, shows the space where the population is and how the infections take place. The panel also shows the graphs to understand how the virus is being spread as per the factors set.

For every cycle-time, for the complete dataset of persons, the below function is being called which is the main algorithm.



```python
def updatePerson(self, ContaminationContact:bool,person: Person, time: int) -> Person:
    quo =False
    if(not person.get_deceased()):
        if(ContaminationContact):
            result = self.getInfectionStatus(person.get_medical_history_scale(), person.get_mask_usage(), person.get_qurantine())
            spread_infection = self.check_will_spread()
            if result:
                if spread_infection:
                    can_infect = np.random.randint(0,self.cu.get_total_to_infect()/2)
                    self.cu.set_total_to_infect(self.cu.get_total_to_infect()-can_infect)
                    self.cu.update_to_infect()
                    person.set_can_infect(can_infect)
                person.set_infected(True)
                person.set_recoveryDays(np.random.random_integers(self.cu.get_days_contageous()))
                quo =True


        # Update movement for the person
        if not person.get_qurantine():
            x, y = self.updateMovement(person.get_x(), person.get_y())
            person.set_x(x)
            person.set_y(y)

        # Update recovery days or demise status if the person is infected
        if person.get_infected() and not person.get_recovered():
            r_days = person.get_recoveryDays()
            if r_days > 0:
                updated_rday = r_days-1 if r_days -1>0 else 0
                person.set_recoveryDays(updated_rday)
            elif r_days <= 0:
                recovered, demise = self.updateDemiseOrRecovered(
                    person.get_medical_history_scale())
                person.set_deceased(demise)
                person.set_recovered(recovered)
                person.set_infected(False)


        # Check if quarantine has been introduced or not and check if the person will quarantine or not
        if self.cu.get_quarantine_introduced_timeline() > time:
            qurantine = person.get_qurantine()
            person.set_qurantine(self.updateQuarantine(qurantine))
        # Check if mask has been introduced or not and check if the person will wear mask or not
        if self.cu.get_mask_introduced_timeline() > time:
            mask = person.get_mask_usage()
            person.set_mask_usage(self.updateMak(mask))
        # Check if vaccine has been introduced or not and check if the person will get vaccinated or not
        if self.cu.get_vaccine_introduced_timeline() > time and not person.get_vaccinated():
            vaccinated = person.get_vaccinated()
            if not vaccinated:
                person.set_vaccinated(vaccinated)
    else:
        person.set_infected(False)
        person.set_qurantine(False)
        person.set_mask_usage(False)

    return person,quo
```

This function does the complete work of updating a person for the current cycle-time.

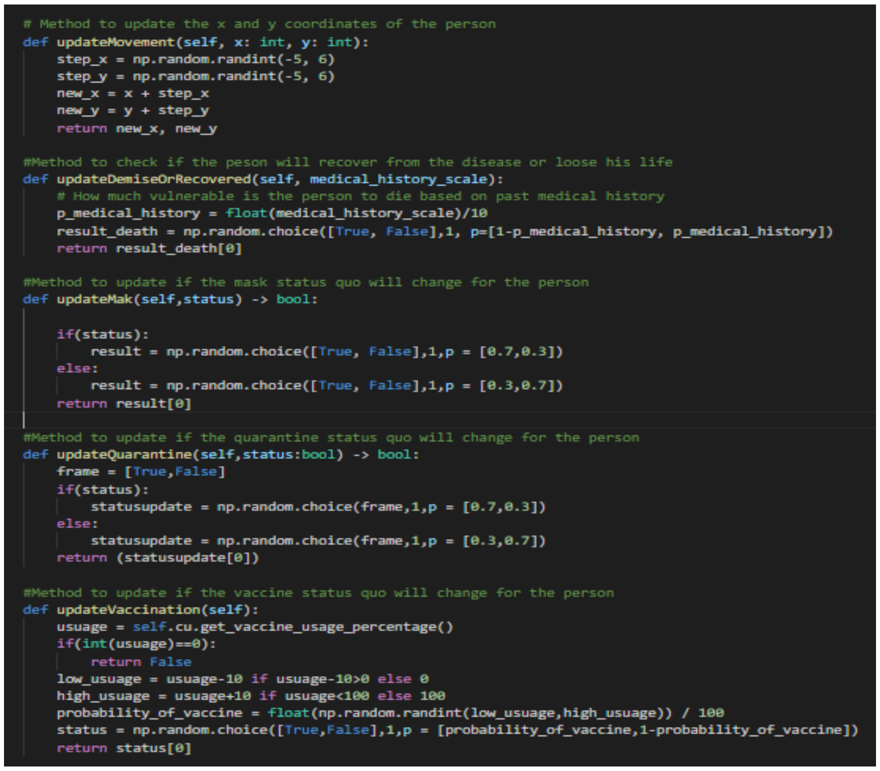
Its computational order:

If the person is alive

If the person is in close contact of any other infected person

```
        If yes : checking their infection status whether the person will get infected
            If yes: infecting and updating how many others the person can infect (K factor)
                Setting randomly for how many days the person will remain infected
    If the person is not quarantined
        Updating the person's movement
    If the person is infected and not yet recovered: Updating the person's recovery day's left
        If recovery days are completed: checking if the person will come under the probability of dead
    If mask has been introduced: checking if the person will wear it
    If quarantine has been introduced: checking if the person will quarantine
    If vaccine has been introduced: checking if the person has taken the vaccination
```

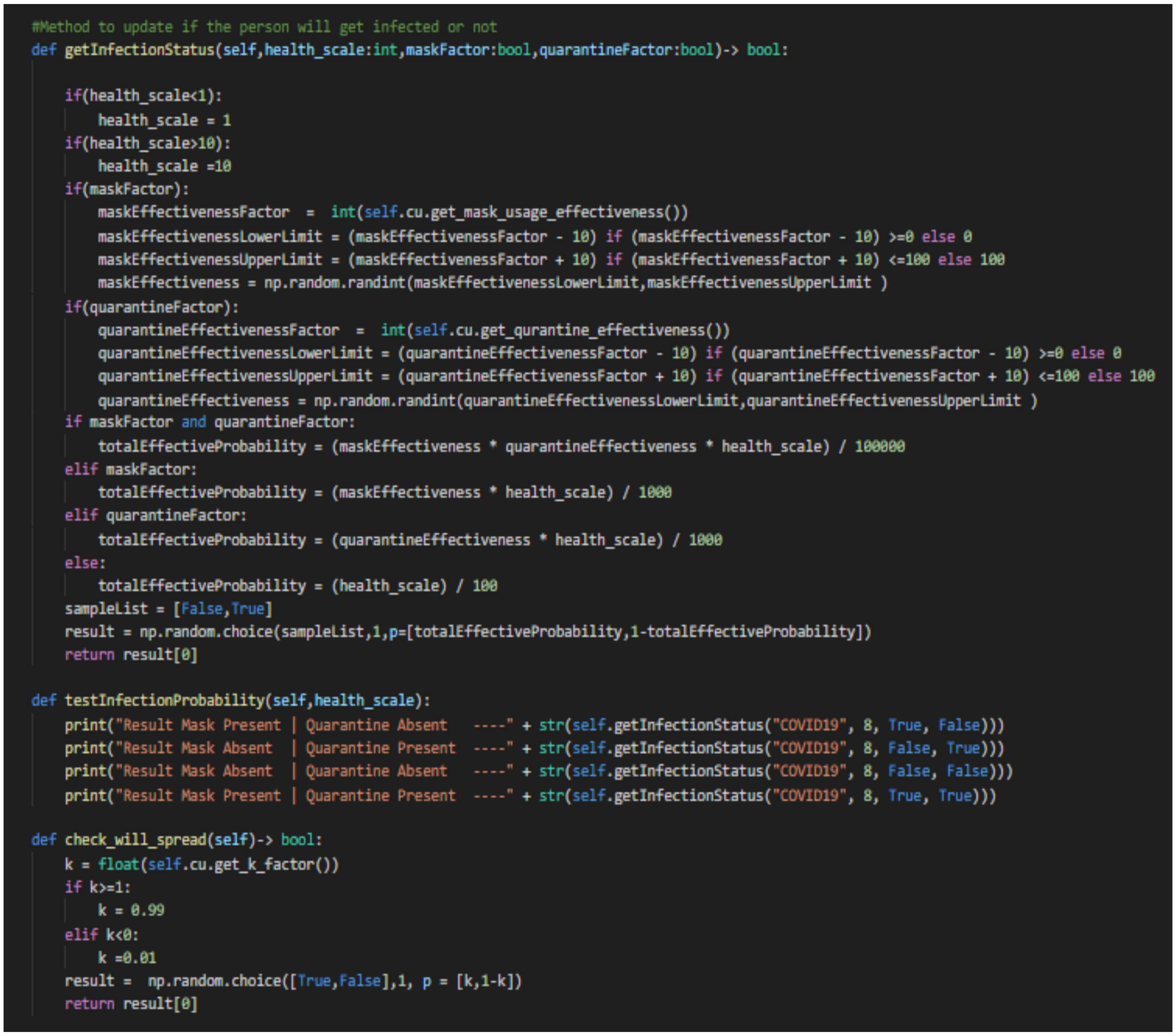
Below are the various functions where we compute all the above-mentioned operations.
Every value required is taken from the config, set by the user, and randomizing the values in its surroundings.
For movement: we are randomly moving the person in the space in between -5 to 5 for both the coordinates
For checking if the person dies of covid, we considered the health scale, how immune the person is at the time of infection, together with the fatality rate of the virus specified in the config.



```python
# Method to update the x and y coordinates of the person
def updateMovement(self, x: int, y: int):
    step_x = np.random.randint(-5, 6)
    step_y = np.random.randint(-5, 6)
    new_x = x + step_x
    new_y = y + step_y
    return new_x, new_y

#Method to check if the peson will recover from the disease or loose his life
def updateDemiseOrRecovered(self, medical_history_scale):
    # How much vulnerable is the person to die based on past medical history
    p_medical_history = float(medical_history_scale)/10
    result_death = np.random.choice([True, False],1, p=[1-p_medical_history, p_medical_history])
    return result_death[0]

#Method to update if the mask status quo will change for the person
def updateMak(self,status) -> bool:

    if(status):
        result = np.random.choice([True, False],1,p = [0.7,0.3])
    else:
        result = np.random.choice([True, False],1,p = [0.3,0.7])
    return result[0]

#Method to update if the quarantine status quo will change for the person
def updateQuarantine(self,status:bool) -> bool:
    frame = [True,False]
    if(status):
        statusupdate = np.random.choice(frame,1,p = [0.7,0.3])
    else:
        statusupdate = np.random.choice(frame,1,p = [0.3,0.7])
    return (statusupdate[0])

#Method to update if the vaccine status quo will change for the person
def updateVaccination(self):
    usage = self.cu.get_vaccine_usage_percentage()
    if(int(usage)==0):
        return False
    low_usuage = usage-10 if usage-10>0 else 0
    high_usuage = usage+10 if usage<100 else 100
    probability_of_vaccine = float(np.random.randint(low_usuage,high_usuage)) / 100
    status = np.random.choice([True,False],1,p = [probability_of_vaccine,1-probability_of_vaccine])
    return status[0]
```

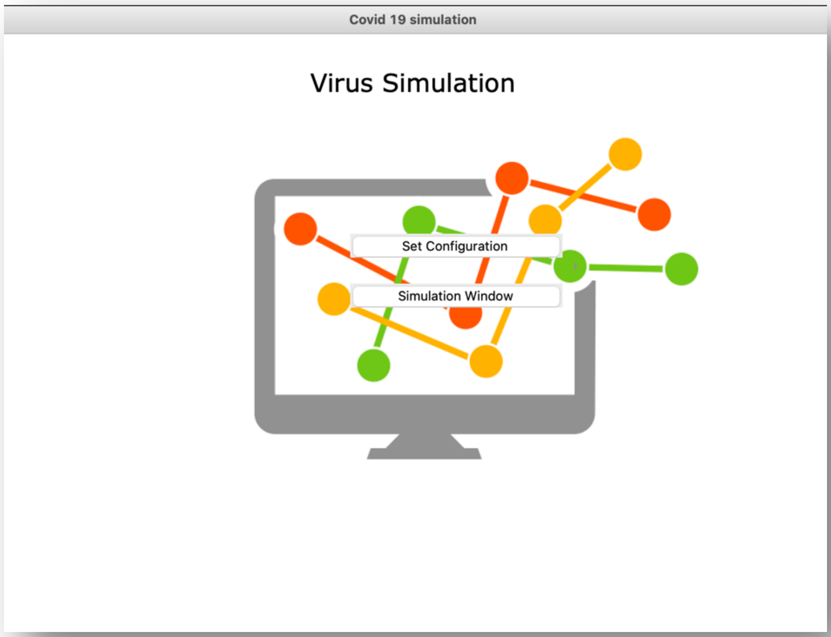
For getting the infection status, we consider all the factors responsible, such as checking if the person has good immunity levels and is healthy, the mask factor, quarantine factor, vaccination factor, all the three separately as well as combination of them.

For checking if the person will spread, we considered the K-factor, the main factor that is responsible for the COVID-19 infection rate, i.e. the super spreaders, which is about 0.1, which says 10% of the infected people are responsible for the overall 90% of the infections.

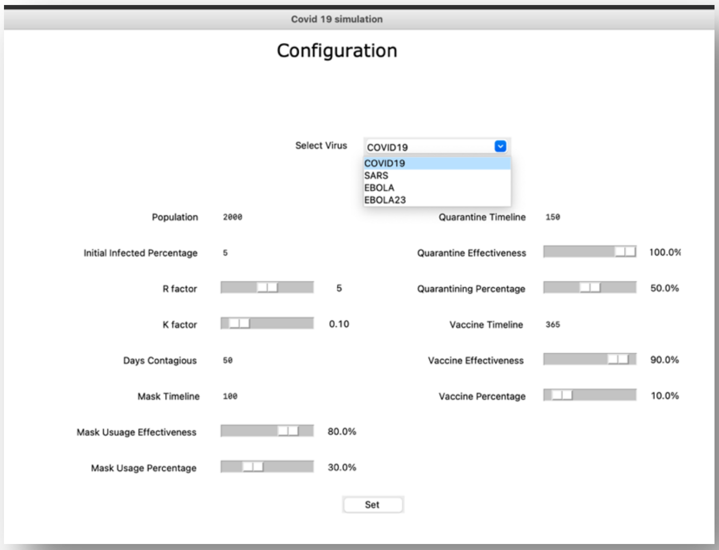


```
#Method to update if the person will get infected or not
def getInfectionStatus(self,health_scale:int,maskFactor:bool,quarantineFactor:bool)-> bool:

    if(health_scale<1):
        health_scale = 1
    if(health_scale>10):
        health_scale =10
    if(maskFactor):
        maskEffectivenessFactor  =  int(self.cu.get_mask_usage_effectiveness())
        maskEffectivenessLowerLimit = (maskEffectivenessFactor - 10) if (maskEffectivenessFactor - 10) >=0 else 0
        maskEffectivenessUpperLimit = (maskEffectivenessFactor + 10) if (maskEffectivenessFactor + 10) <=100 else 100
        maskEffectiveness = np.random.randint(maskEffectivenessLowerLimit,maskEffectivenessUpperLimit )
    if(quarantineFactor):
        quarantineEffectivenessFactor  =  int(self.cu.get_qurantine_effectiveness())
        quarantineEffectivenessLowerLimit = (quarantineEffectivenessFactor - 10) if (quarantineEffectivenessFactor - 10) >=0 else 0
        quarantineEffectivenessUpperLimit = (quarantineEffectivenessFactor + 10) if (quarantineEffectivenessFactor + 10) <=100 else 100
        quarantineEffectiveness = np.random.randint(quarantineEffectivenessLowerLimit,quarantineEffectivenessUpperLimit )
    if maskFactor and quarantineFactor:
        totalEffectiveProbability = (maskEffectiveness * quarantineEffectiveness * health_scale) / 100000
    elif maskFactor:
        totalEffectiveProbability = (maskEffectiveness * health_scale) / 1000
    elif quarantineFactor:
        totalEffectiveProbability = (quarantineEffectiveness * health_scale) / 1000
    else:
        totalEffectiveProbability = (health_scale) / 100
    sampleList = [False,True]
    result = np.random.choice(sampleList,1,p=[totalEffectiveProbability,1-totalEffectiveProbability])
    return result[0]

def testInfectionProbability(self,health_scale):
    print("Result Mask Present | Quarantine Absent   ----" + str(self.getInfectionStatus("COVID19", 8, True, False)))
    print("Result Mask Absent  | Quarantine Present  ----" + str(self.getInfectionStatus("COVID19", 8, False, True)))
    print("Result Mask Absent  | Quarantine Absent   ----" + str(self.getInfectionStatus("COVID19", 8, False, False)))
    print("Result Mask Present | Quarantine Present  ----" + str(self.getInfectionStatus("COVID19", 8, True, True)))

def check_will_spread(self)-> bool:
    k = float(self.cu.get_k_factor())
    if k>=1:
        k = 0.99
    elif k<0:
        k =0.01
    result =  np.random.choice([True,False],1, p = [k,1-k])
    return result[0]
```

OUTPUT



2- Entry Screen

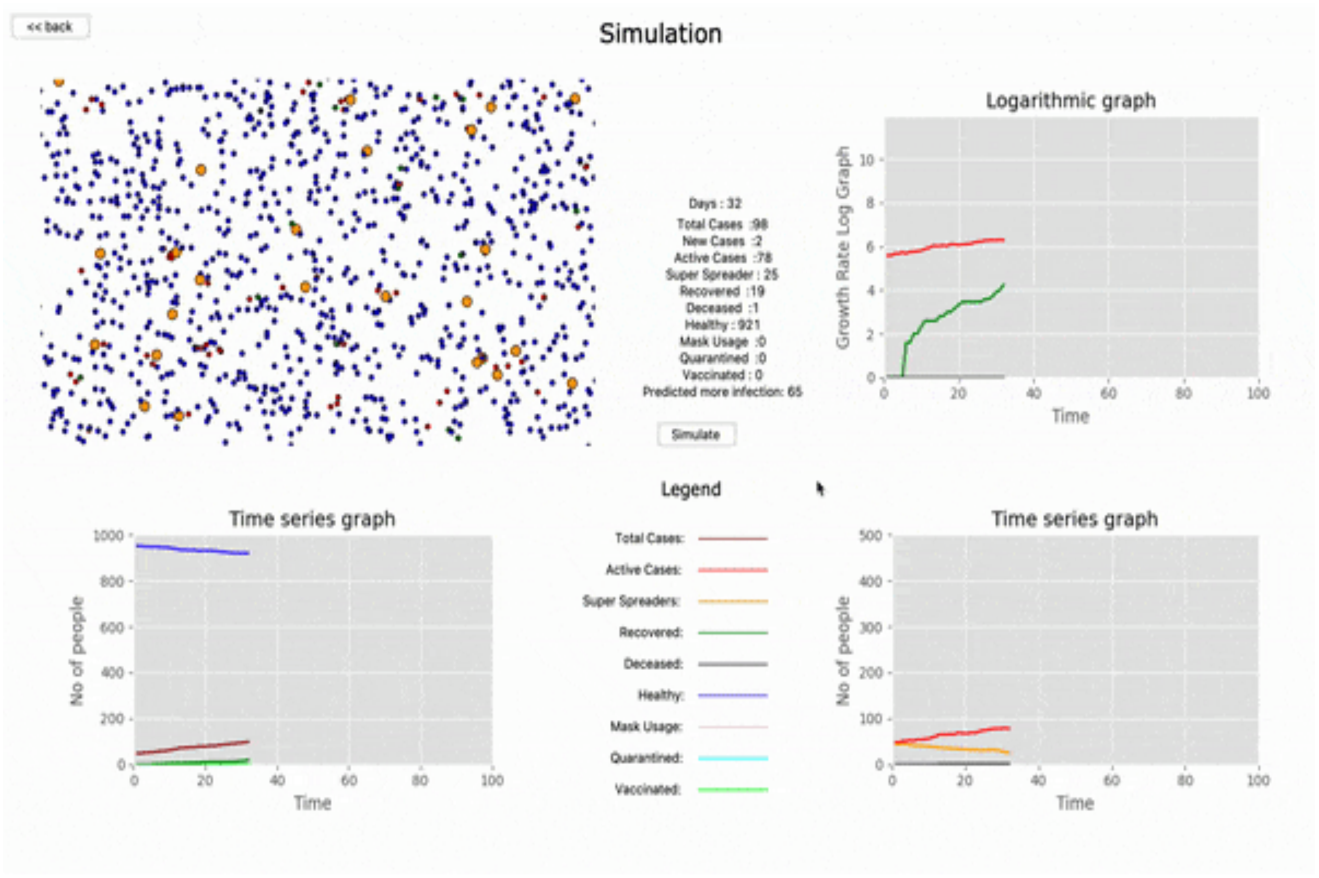


Configuration

Select Virus: COVID19 (COVID19, SARS, EBOLA, EBOLA23)

Population	2000	Quarantine Timeline	150
Initial Infected Percentage	5	Quarantine Effectiveness	100.0%
R factor	5	Quarantining Percentage	50.0%
K factor	0.10	Vaccine Timeline	365
Days Contagious	50	Vaccine Effectiveness	90.0%
Mask Timeline	100	Vaccine Percentage	10.0%
Mask Usuage Effectiveness	80.0%		
Mask Usage Percentage	30.0%		

3- Configuration Screen



```
Days : 32
Total Cases :98
New Cases :2
Active Cases :78
Super Spreader : 25
Recovered :19
Deceased :1
Healthy : 921
Mask Usage :0
Quarantined :0
Vaccinated : 0
Predicted more infection: 65
```

4- Simulation Screen

ANALYSIS

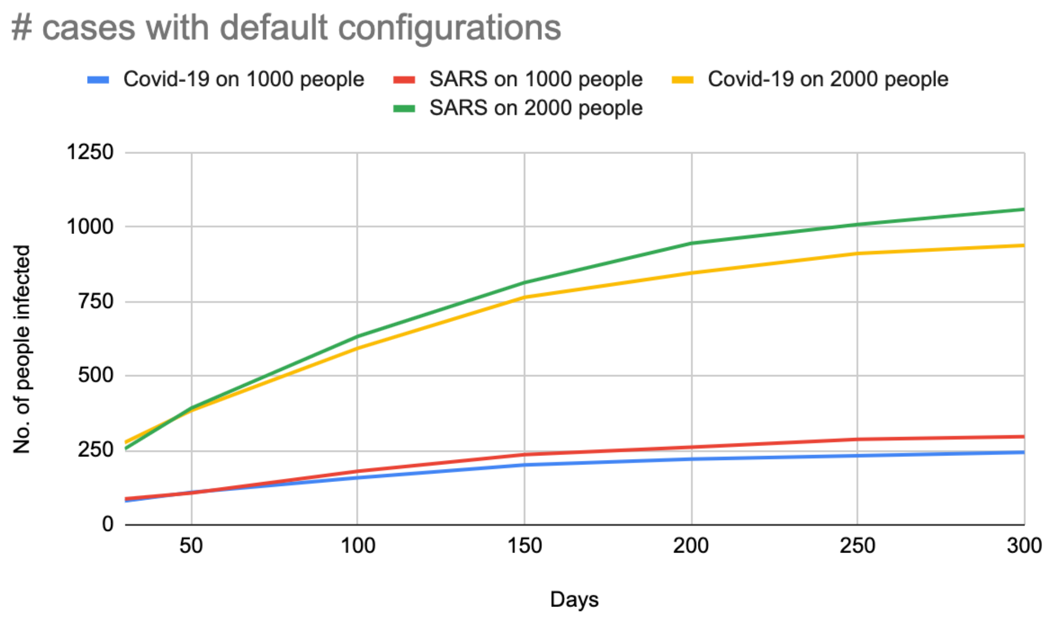
Performance:

Time Complexity : The alogoritghm worst case complexity is O(N*T) where N is the number of people and T is the time the simulation is run.
Space Complexity : The algorithm complexity is O(N+T) where N is the number people to simulate and T is time the simulation is run. Additional T value is needed to store the state of N at time T for plotting graphs

Simulation:

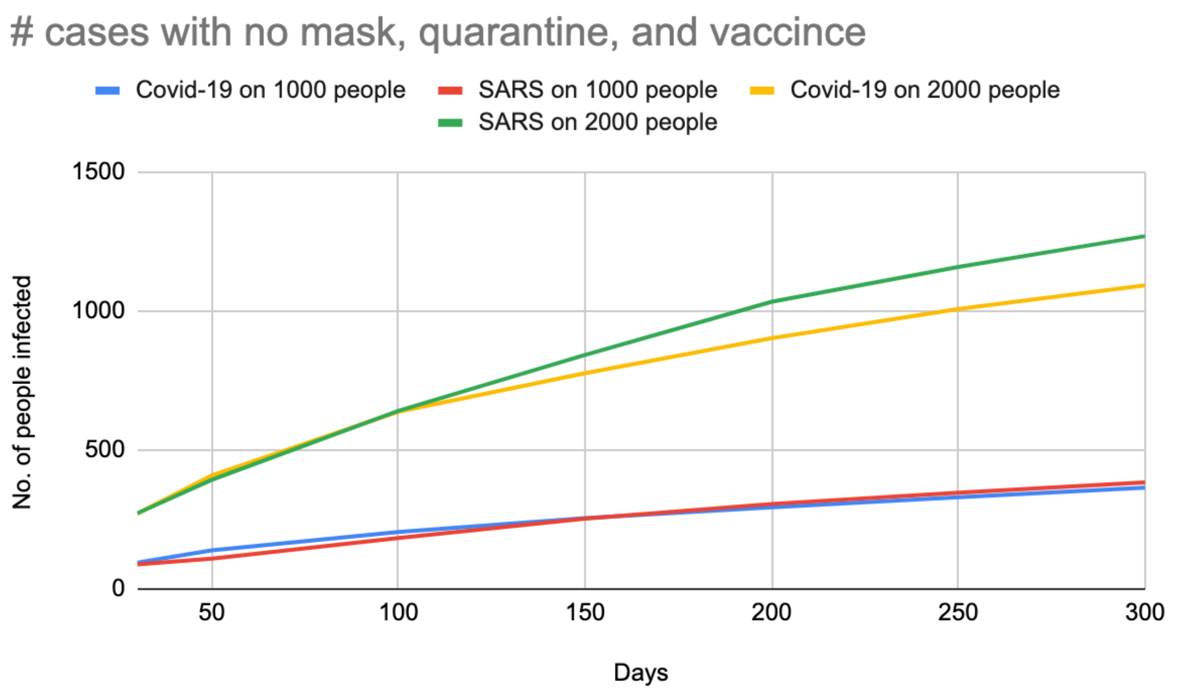
Case 1 : Mask, quarantine and vaccine are present

Days	Covid-19 on 1000 people	SARS on 1000 people	Covid-19 on 2000 people	SARS on 2000 people
30	80	87	276	255
50	109	107	384	392
100	158	180	593	633
150	201	236	764	813
200	221	261	845	945
250	232	287	911	1008
300	243	296	938	1059



Case 2 : No Mask, No quarantine and No vaccine are present

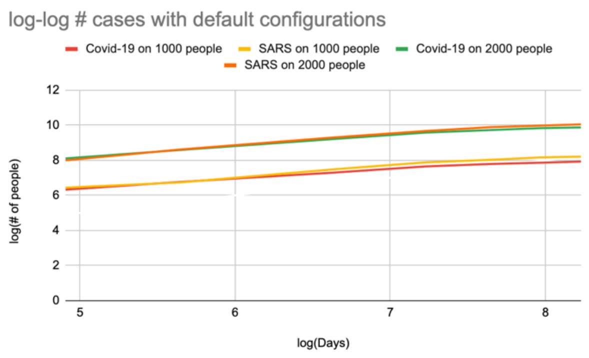
Days	Covid-19 on 1000 people	SARS on 1000 people	Covid-19 on 2000 people	SARS on 2000 people
30	95	88	271	272
50	139	109	408	393
100	205	183	638	641
150	255	253	776	842
200	294	306	902	1033
250	330	346	1007	1158
300	364	383	1092	1269

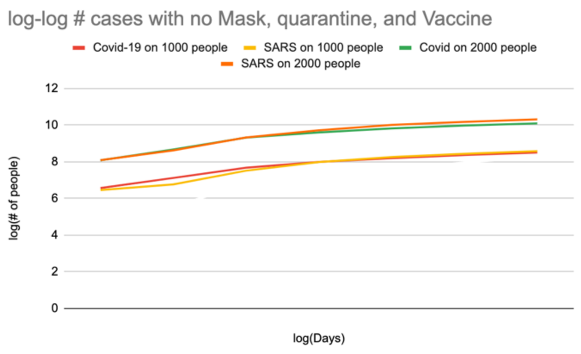


CONCLUSION

For time T(Days) and number of infected population N

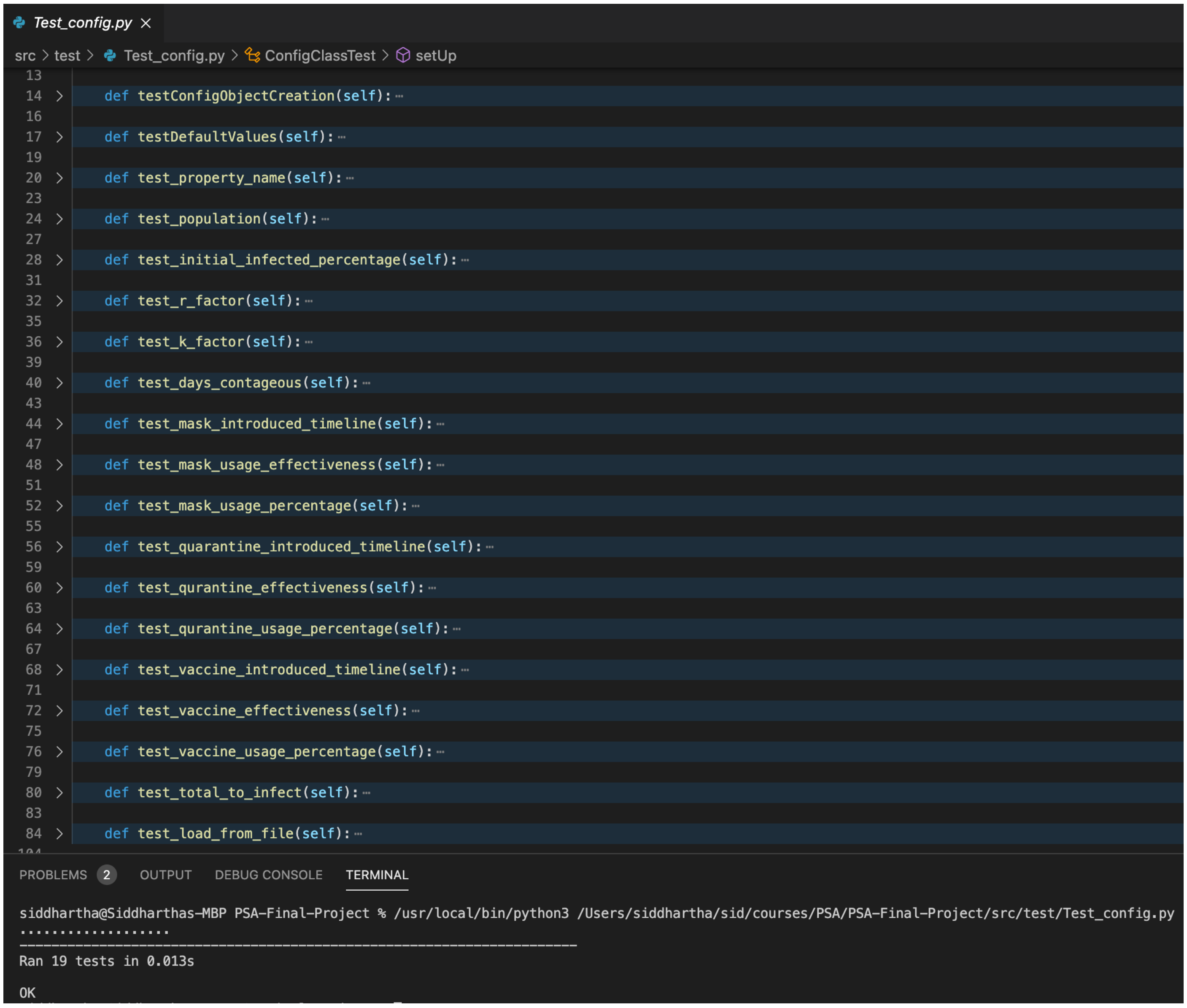
Disease	With Mask, Quarantine and Vaccination	Without Mask, Quarantine and Vaccination
Covid19 – 1000 Person	N ≅ 1.087670851 * T	N ≅ 1.145563034* T
Covid19 – 2000 Person	N ≅ 1.114069835* T	N ≅ 1.133301742* T
SARS – 1000 Person	N ≅ 1.270149541* T	N ≅ 1.382468493* T
SARS – 2000Person	N ≅ 1.280048267 * T	N ≅ 1.394761819* T





UNIT TESTING

1) Test_Config.py



Test_config.py ×

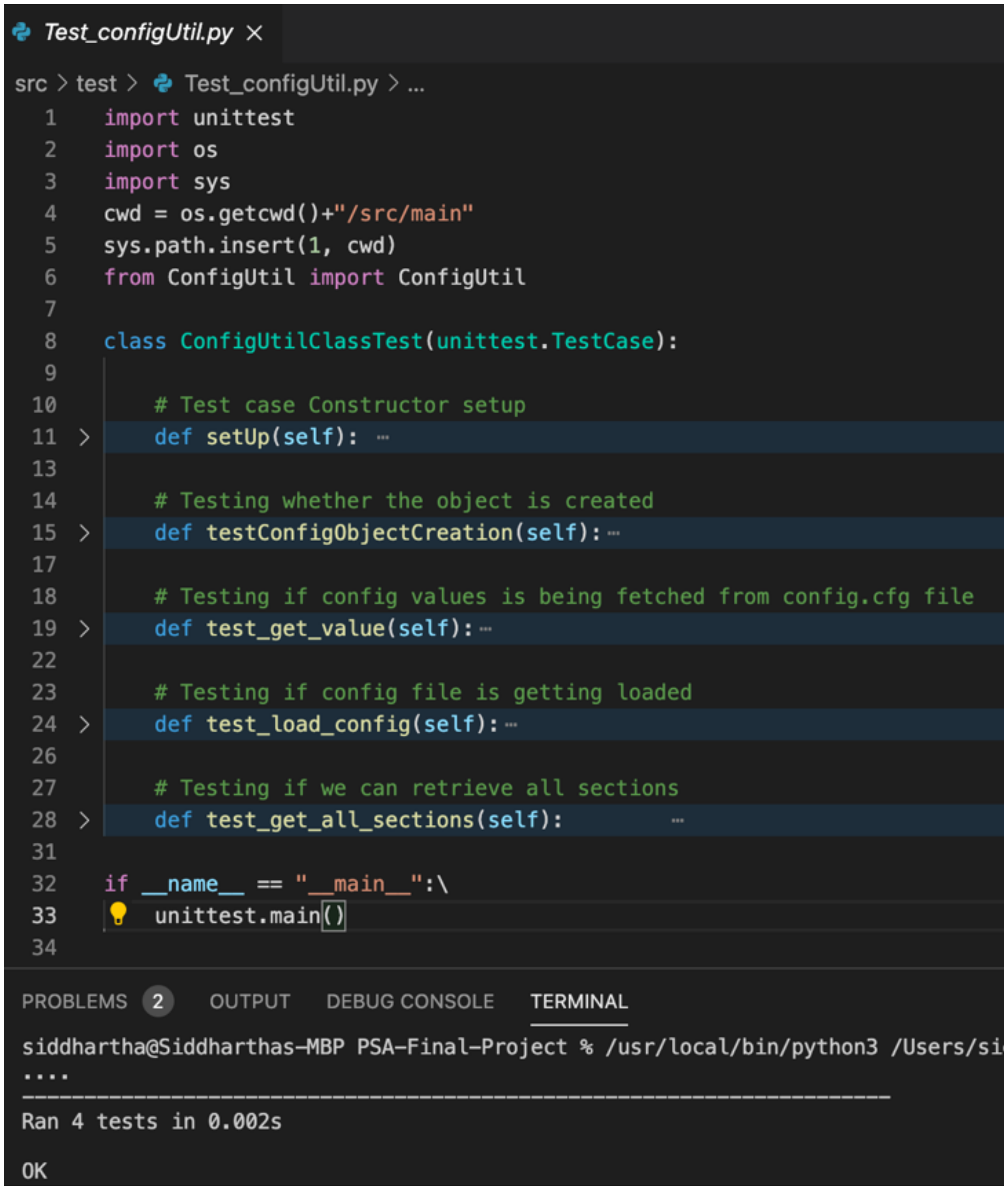
src > test > Test_config.py > ConfigClassTest > setUp

```
    def testConfigObjectCreation(self):⋯

    def testDefaultValues(self):⋯

    def test_property_name(self):⋯

    def test_population(self):⋯

    def test_initial_infected_percentage(self):⋯

    def test_r_factor(self):⋯

    def test_k_factor(self):⋯

    def test_days_contageous(self):⋯

    def test_mask_introduced_timeline(self):⋯

    def test_mask_usage_effectiveness(self):⋯

    def test_mask_usage_percentage(self):⋯

    def test_quarantine_introduced_timeline(self):⋯

    def test_qurantine_effectiveness(self):⋯

    def test_qurantine_usage_percentage(self):⋯

    def test_vaccine_introduced_timeline(self):⋯

    def test_vaccine_effectiveness(self):⋯

    def test_vaccine_usage_percentage(self):⋯

    def test_total_to_infect(self):⋯

    def test_load_from_file(self):⋯
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

```
siddhartha@Siddharthas-MBP PSA-Final-Project % /usr/local/bin/python3 /Users/siddhartha/sid/courses/PSA/PSA-Final-Project/src/test/Test_config.py
...................
----------------------------------------------------------------------
Ran 19 tests in 0.013s

OK
```

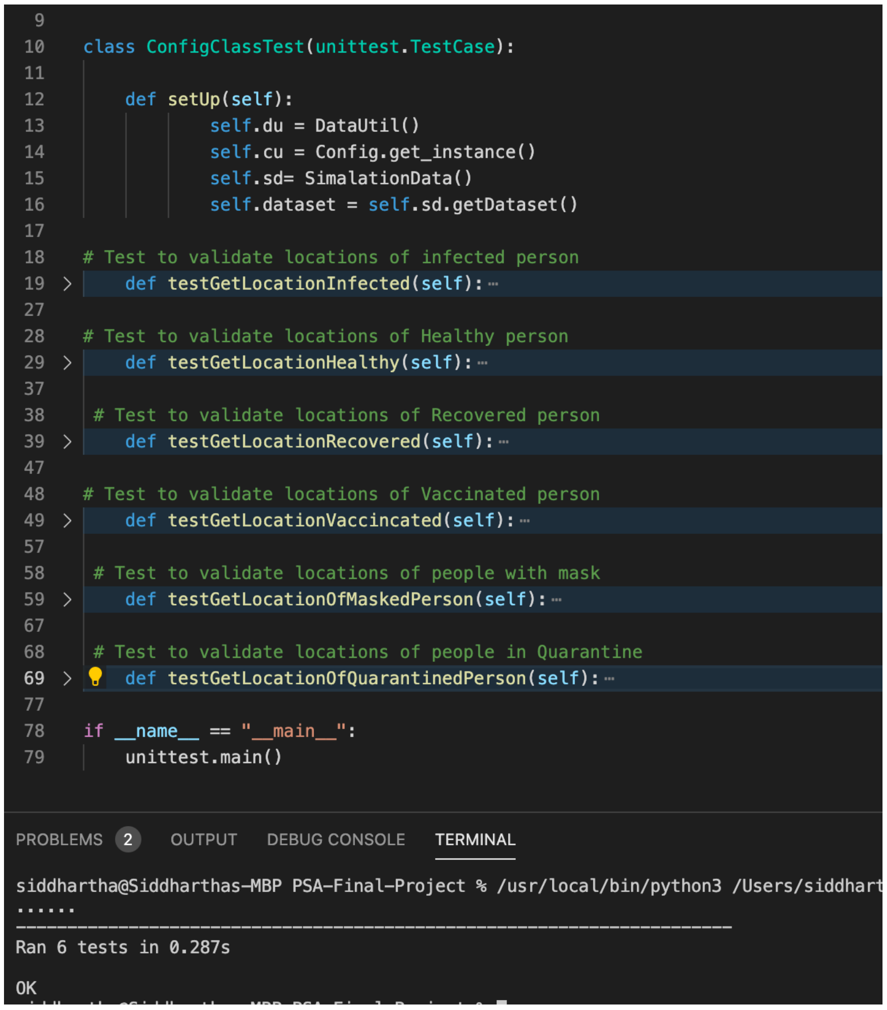
2) Test_ConfigUtil.py



```python
import unittest
import os
import sys
cwd = os.getcwd()+"/src/main"
sys.path.insert(1, cwd)
from ConfigUtil import ConfigUtil

class ConfigUtilClassTest(unittest.TestCase):

    # Test case Constructor setup
    def setUp(self): ...

    # Testing whether the object is created
    def testConfigObjectCreation(self):...

    # Testing if config values is being fetched from config.cfg file
    def test_get_value(self):...

    # Testing if config file is getting loaded
    def test_load_config(self):...

    # Testing if we can retrieve all sections
    def test_get_all_sections(self):        ...

if __name__ == "__main__":\
    unittest.main()

```

```
siddhartha@Siddharthas-MBP PSA-Final-Project % /usr/local/bin/python3 /Users/si
....
----------------------------------------------------------------------
Ran 4 tests in 0.002s

OK
```

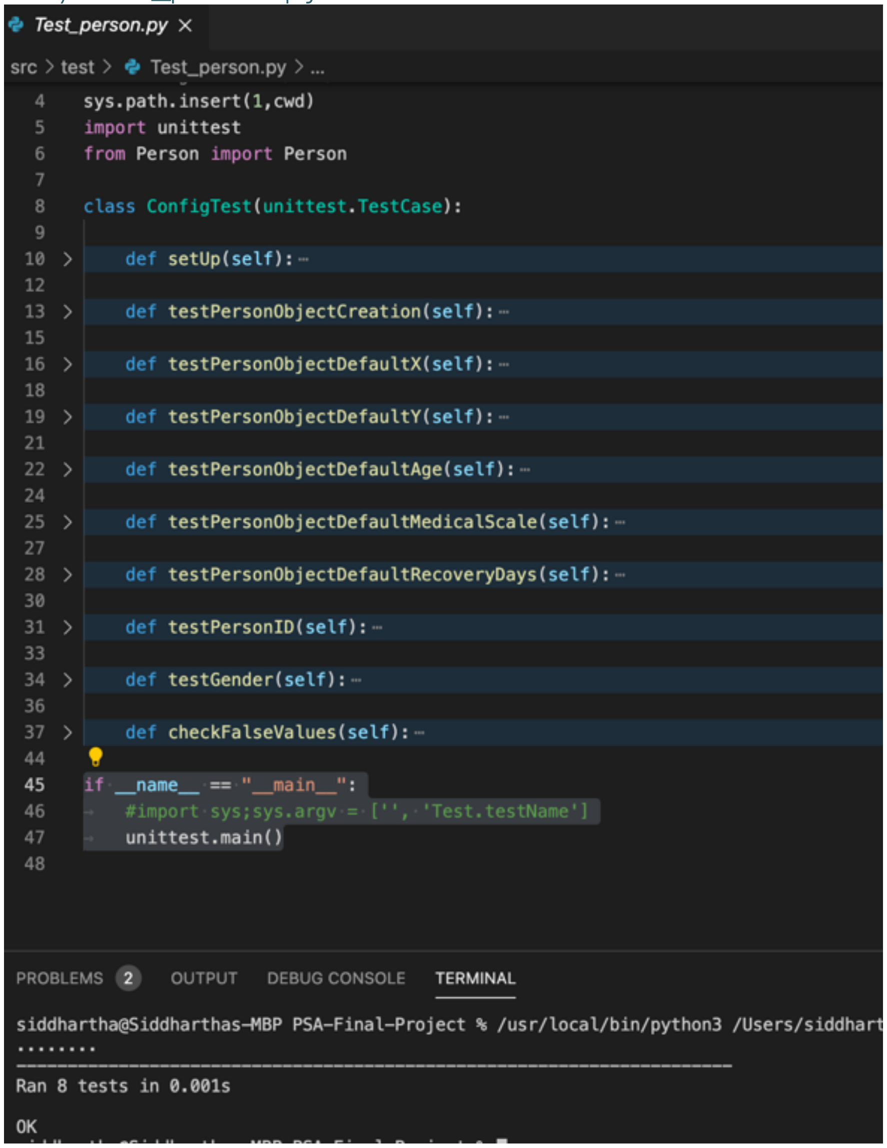
3) Test_DataUtil.py



```python

class ConfigClassTest(unittest.TestCase):

    def setUp(self):
            self.du = DataUtil()
            self.cu = Config.get_instance()
            self.sd= SimalationData()
            self.dataset = self.sd.getDataset()

# Test to validate locations of infected person
    def testGetLocationInfected(self):⋯

# Test to validate locations of Healthy person
    def testGetLocationHealthy(self):⋯

 # Test to validate locations of Recovered person
    def testGetLocationRecovered(self):⋯

# Test to validate locations of Vaccinated person
    def testGetLocationVaccincated(self):⋯

 # Test to validate locations of people with mask
    def testGetLocationOfMaskedPerson(self):⋯

 # Test to validate locations of people in Quarantine
    def testGetLocationOfQuarantinedPerson(self):⋯

if __name__ == "__main__":
    unittest.main()
```

```
siddhartha@Siddharthas-MBP PSA-Final-Project % /usr/local/bin/python3 /Users/siddhart
......
----------------------------------------------------------------------
Ran 6 tests in 0.287s

OK
```

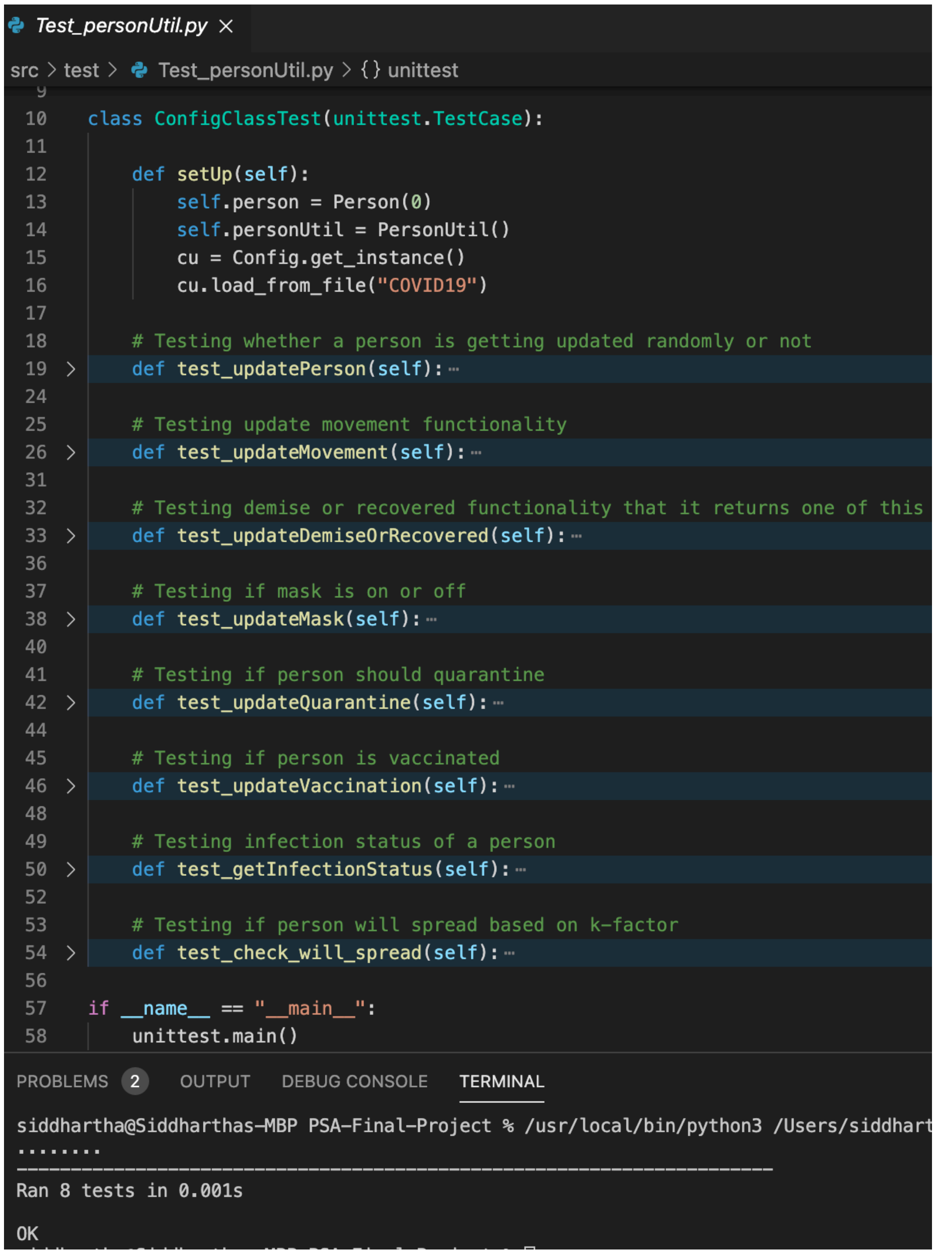
4) Test_person.py



```
sys.path.insert(1,cwd)
import unittest
from Person import Person

class ConfigTest(unittest.TestCase):

    def setUp(self):…

    def testPersonObjectCreation(self):…

    def testPersonObjectDefaultX(self):…

    def testPersonObjectDefaultY(self):…

    def testPersonObjectDefaultAge(self):…

    def testPersonObjectDefaultMedicalScale(self):…

    def testPersonObjectDefaultRecoveryDays(self):…

    def testPersonID(self):…

    def testGender(self):…

    def checkFalseValues(self):…

if __name__ == "__main__":
    #import sys;sys.argv = ['', 'Test.testName']
    unittest.main()
```

```
siddhartha@Siddharthas-MBP PSA-Final-Project % /usr/local/bin/python3 /Users/siddhart
........
----------------------------------------------------------------------
Ran 8 tests in 0.001s

OK
```

5) Test_PersonUtil.py



Test_personUtil.py

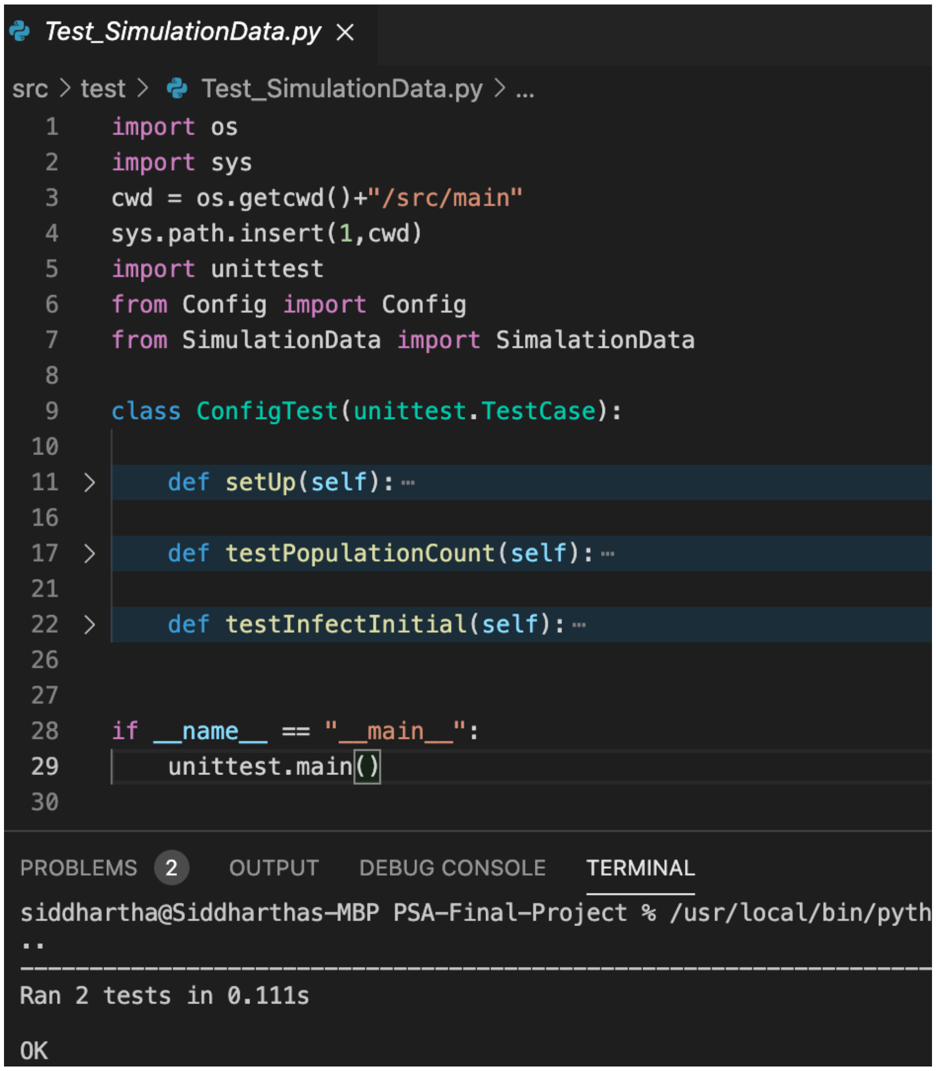
src > test > Test_personUtil.py > {} unittest

```python
class ConfigClassTest(unittest.TestCase):

    def setUp(self):
        self.person = Person(0)
        self.personUtil = PersonUtil()
        cu = Config.get_instance()
        cu.load_from_file("COVID19")

    # Testing whether a person is getting updated randomly or not
    def test_updatePerson(self):⋯

    # Testing update movement functionality
    def test_updateMovement(self):⋯

    # Testing demise or recovered functionality that it returns one of this
    def test_updateDemiseOrRecovered(self):⋯

    # Testing if mask is on or off
    def test_updateMask(self):⋯

    # Testing if person should quarantine
    def test_updateQuarantine(self):⋯

    # Testing if person is vaccinated
    def test_updateVaccination(self):⋯

    # Testing infection status of a person
    def test_getInfectionStatus(self):⋯

    # Testing if person will spread based on k-factor
    def test_check_will_spread(self):⋯

if __name__ == "__main__":
    unittest.main()
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

```
siddhartha@Siddharthas-MBP PSA-Final-Project % /usr/local/bin/python3 /Users/siddhart
........
----------------------------------------------------------------------
Ran 8 tests in 0.001s

OK
```

6) Test_SimulationData.py



Test_SimulationData.py ×

src > test > Test_SimulationData.py > ...

```python
import os
import sys
cwd = os.getcwd()+"/src/main"
sys.path.insert(1,cwd)
import unittest
from Config import Config
from SimulationData import SimalationData

class ConfigTest(unittest.TestCase):

    def setUp(self):...

    def testPopulationCount(self):...

    def testInfectInitial(self):...


if __name__ == "__main__":
    unittest.main()

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

```
siddhartha@Siddharthas-MBP PSA-Final-Project % /usr/local/bin/pyth
..
----------------------------------------------------------------------
Ran 2 tests in 0.111s

OK
```

REREFENCES

1) https://www.doh.wa.gov/Portals/1/Documents/1600/coronavirus/WearAClothFaceCovering.pdf

2) https://royalsociety.org/-/media/policy/projects/set-c/set-covid-19-R-estimates.pdf

3) https://www.healthline.com/health/r-nought-reproduction-number#covid-19-r-0

4) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2851497/

5) https://www.cdc.gov/sars/clinical/respirators.html

6) https://jbiomedsci.biomedcentral.com/articles/10.1186/s12929-020-00695-2

7) https://sph.umich.edu/pursuit/2020posts/how-scientists-quantify-outbreaks.html