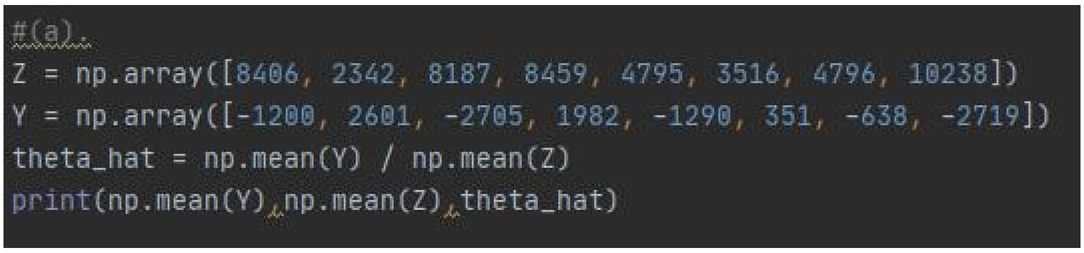
1.

(a) Calculate the plug-in estimate of θ, which is equal to $\hat{\theta} = \overline{Y} / \overline{Z}$. [7 Marks]

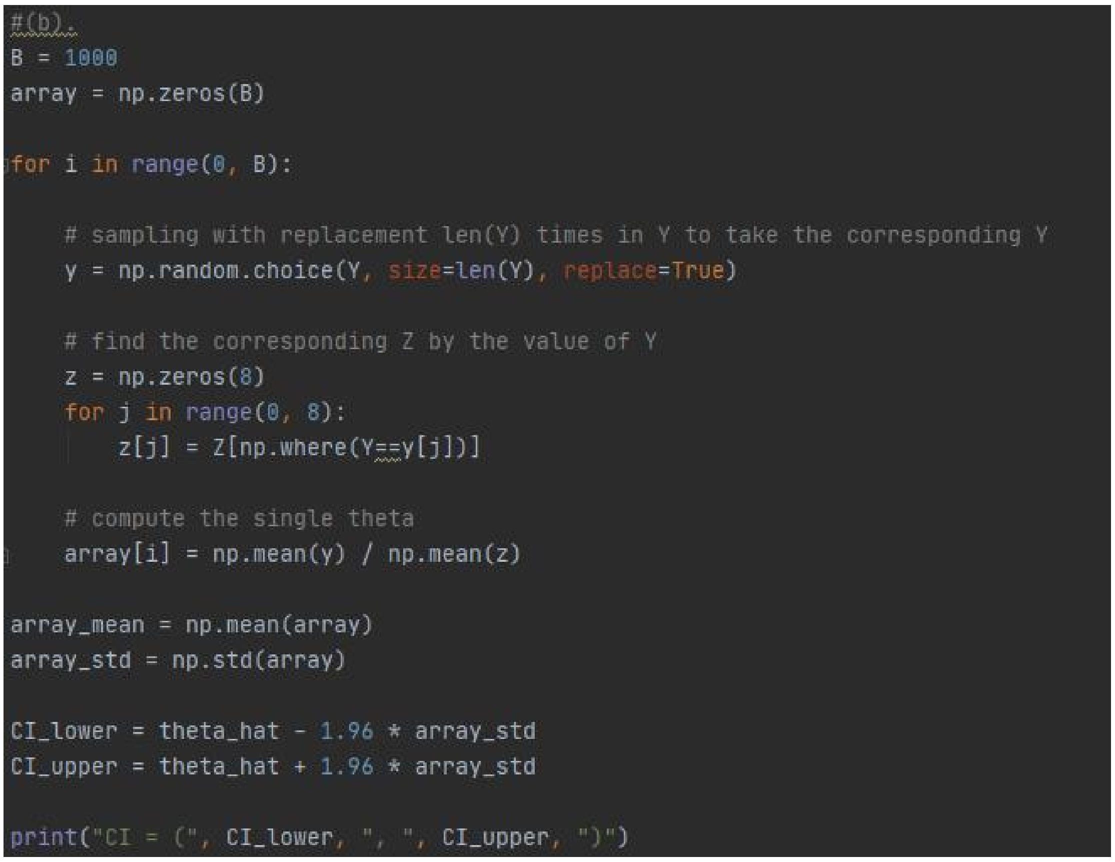


```
#(a).
Z = np.array([8406, 2342, 8187, 8459, 4795, 3516, 4796, 10238])
Y = np.array([-1200, 2601, -2705, 1982, -1290, 351, -638, -2719])
theta_hat = np.mean(Y) / np.mean(Z)
print(np.mean(Y),np.mean(Z),theta_hat)
```

According to the question, we could know that Z = old - placebo, and Y = new - old. Thus, base on the table, we could get $\overline{Z} =$ 6342.375 and $\overline{Y} =$ -452.25.

After that, according to the formula given by question, $\hat{\theta} = \overline{Y} / \overline{Z}$, we could calculate the plug-in estimate of θ is -0.07130609590256017.

(b) Using the bootstrap method with B = 1000 replications, calculate the 95% confidence interval. Compare the obtained interval with the desired quantity: $|\theta| \leq 0.20$. What is your conclusion? [13 Marks]



```
#(b).
B = 1000
array = np.zeros(B)

for i in range(0, B):

    # sampling with replacement len(Y) times in Y to take the corresponding Y
    y = np.random.choice(Y, size=len(Y), replace=True)

    # find the corresponding Z by the value of Y
    z = np.zeros(8)
    for j in range(0, 8):
        z[j] = Z[np.where(Y==y[j])]

    # compute the single theta
    array[i] = np.mean(y) / np.mean(z)

array_mean = np.mean(array)
array_std = np.std(array)

CI_lower = theta_hat - 1.96 * array_std
CI_upper = theta_hat + 1.96 * array_std

print("CI = (", CI_lower, ", ", CI_upper, ")")
```

Using the bootstrap method with B = 1000 replications, calculate the 95% confidence interval is (-0.27463785233943927 , 0.13202566053431897).

The absolute value of the lower limit of the confidence interval is 0.27463785233943927 which is greater than $|\theta| \leq 0.20$. Therefore, we could find that the new medication doesn’t have “bioequivalence”, which means that the new drug is substantially different than the current treatment.

2.

(a) Give the problem summary and describe the project objective. [20 Marks]

the problem summary

The scenario given by question is a discrete-event system (DES), which makes us to simulate the package processing of post office. The discrete-event systems (DES), are those systems in which the state variables change instantaneously through jumps at discrete points in time.

Such system consists of related components or elements that form a complex organization. In the scenario given by question, we could find out that the post-office workers and packages are considered as system components.

The DES in this scenario is a M/M/C (multiple server queue model) because there are more than one queue in this DES. This case sets the time limitation is 2000.

In addition, these components might have a certain characteristic called attributes. The attributes can have both logical and numeric values. For example, the available workers and individual skills can be important component attributes.

Finally, an interaction between the system components affects the system current state. For example, a package which is leaving the system after processing affects the number of available workers and the waiting queue length. In this way, component interactions may cause a complex system behavior over time.

the project objective

We could consider the project objective in this simulation consists of 9 post-office workers and n packages.

At first, all packages are waiting in the queue. As soon as a package arrives, it needs to be processed by a worker.

We could also assume that there is an unfortunate case that all post-office workers are busy, a package enters a waiting queue.

The next available worker will fetch a package from the queue and process it.

If there are no package in the queue, the worker will non-utilized until next package arrives.

In such problem, we will be interested in the system utilization. Namely, how many packages are processed and how many post-office workers are busy. We are generally concerned with the average performance over a long period of time.

(b) Give a specification of variables used in the simulation study. In addition, show a diagram that describes the project dynamics. [10 Marks]

A specification of variables

There are two parts of variables in this simulation study. We are interested in the number of utilized packages and post-office workers, the following holds.

(1) The system state $\{X_t\}_{\{t\geq 0\}}$ is $X_t = (X_{m,t}, X_{r,t}, W)$, where $X_{m,t}$ and $X_{r,t}$ stands for the number of utilized packages and workers respectively. W represents the waiting queue. There are two waiting queues in this study, the regular and the priority queues. The worker will fetch a package from a priority queue. The simulation will stop when t=T=2000.

(2) The system components are packages and post-office workers. In addition, there are two possible events: package arrival and package departure. Note that the system state doesn't change between the events.

(3) We could assume that the life and the process times are distributed according to Exp(λ) and Exp(μ), respectively.

(4) An essential data structure is the waiting queue, in which there are waiting packages in both the regular and the priority queues until the first worker becomes available. And then, the worker will fetch a package from a priority queue. If all workers are busy with package processing, a newly arrived regular (or priority) package enters a regular (or a priority) waiting queue. As soon as a worker finishes a package processing, she either takes a new package from the waiting queues or remains idle if there are no packages waiting.We might need additional flags for the post-office worker(free/busy).

In order to visually see all the variables, I've made a table listed below:

System components	Packages, post-office workers
System state	the number of packages in each queue, the regular and the priority queues, and the current time t. Every event occurring in this system requires to change the state, as changing the number of packages in the correspond queue, and the time of the event needs to be recorded. The simulation will stop when t=T=2000.
System event	Package arrival and package departure from each queue. Each event occurring

	will change the system state. The event type also represents the sub- activity which should enter into in the programming.
System distribution	The processing time of each post-office worker is exponentially distributed. Regular and priority packages arrive to a post office according to a Poisson process.
System queue	There are a priority queue and a regular queue for each post-office worker. If all workers are busy with package processing, a newly arrived regular (or priority) package enters a regular (or a priority) waiting queue. As soon as a worker finishes a package processing, she either takes a new package from the waiting queues or remains idle if there are no packages waiting. In the programming, the regular queue records the number of packages waiting for each worker.
Service flags	Flag represents whether a posy-office worker is busy or not.

Diagram 1: waiting queue

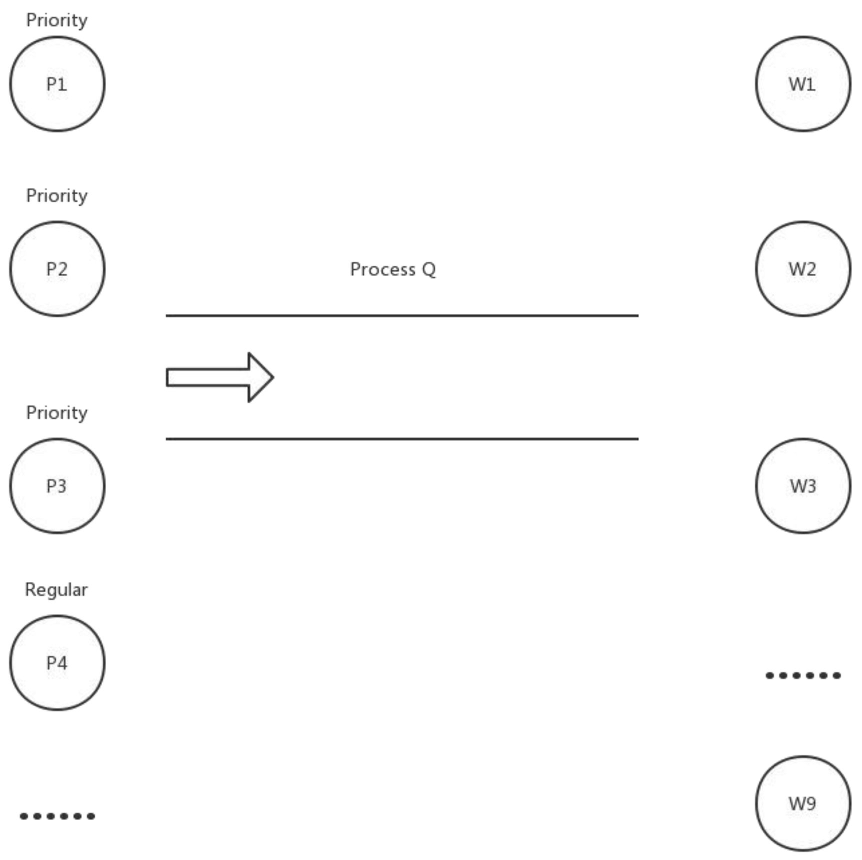

Priority
P1
W1
Priority
P2
Process Q
W2
Priority
P3
W3
Regular
P4
W9

Figure1

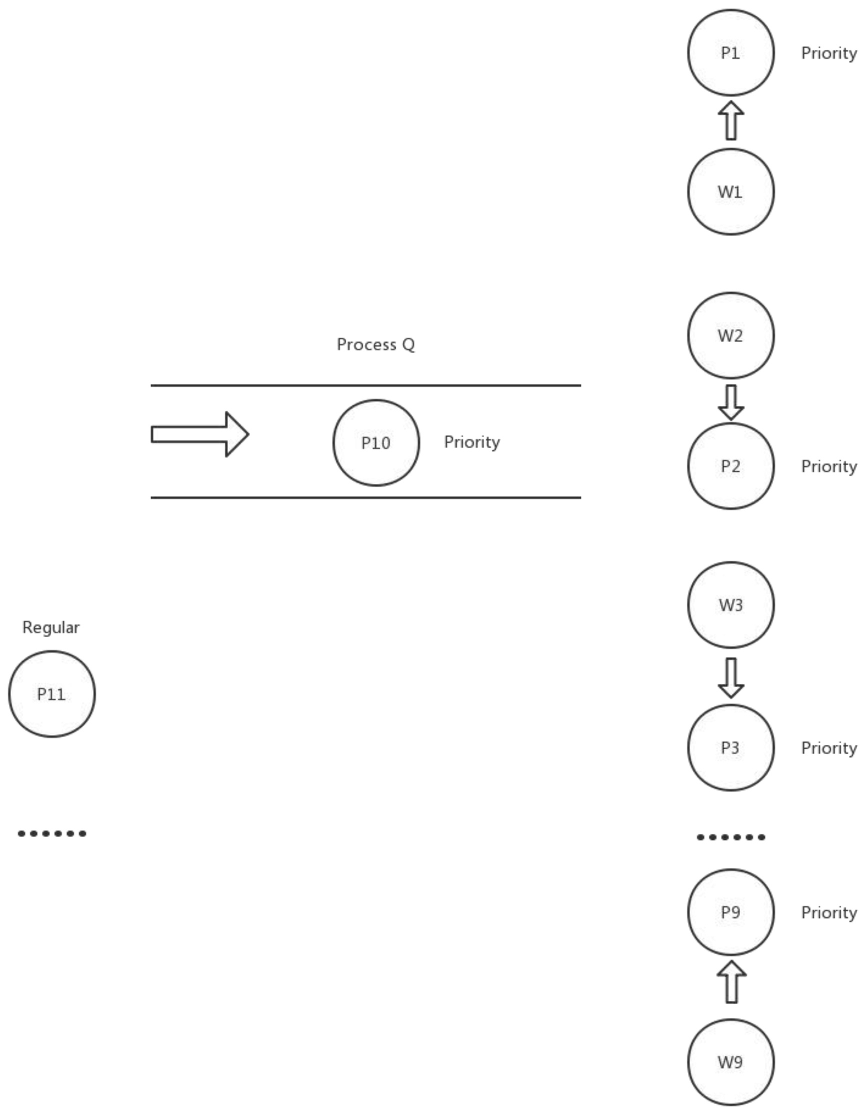


Figure2

Figure: Figure 1 shows the packages system with n packages which are consists of regular and priority packages and 9 post-office workers. Figure 2 shows the same system after the priority package1 to 9 arrived. As the question said the worker will fetch a priority package from the waiting queue first if priority and regular packages are both in the waiting queue. Note that in this case all post-office workers are busy with priority packages 1 to 9, while the priority package 10 is located in the process queue and regular package 11 is not be fetched by any worker.

Diagram 2: general DES process flow

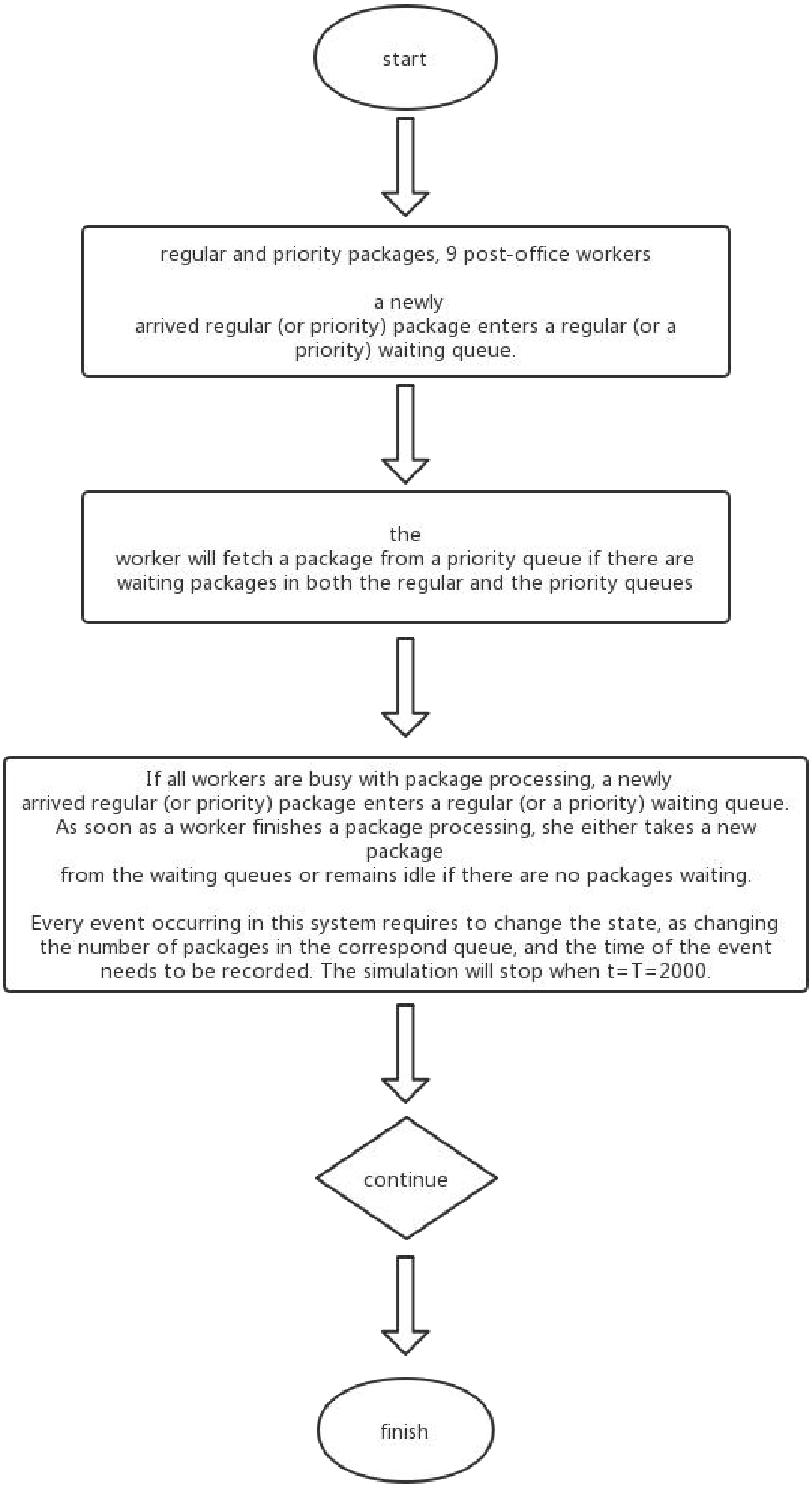
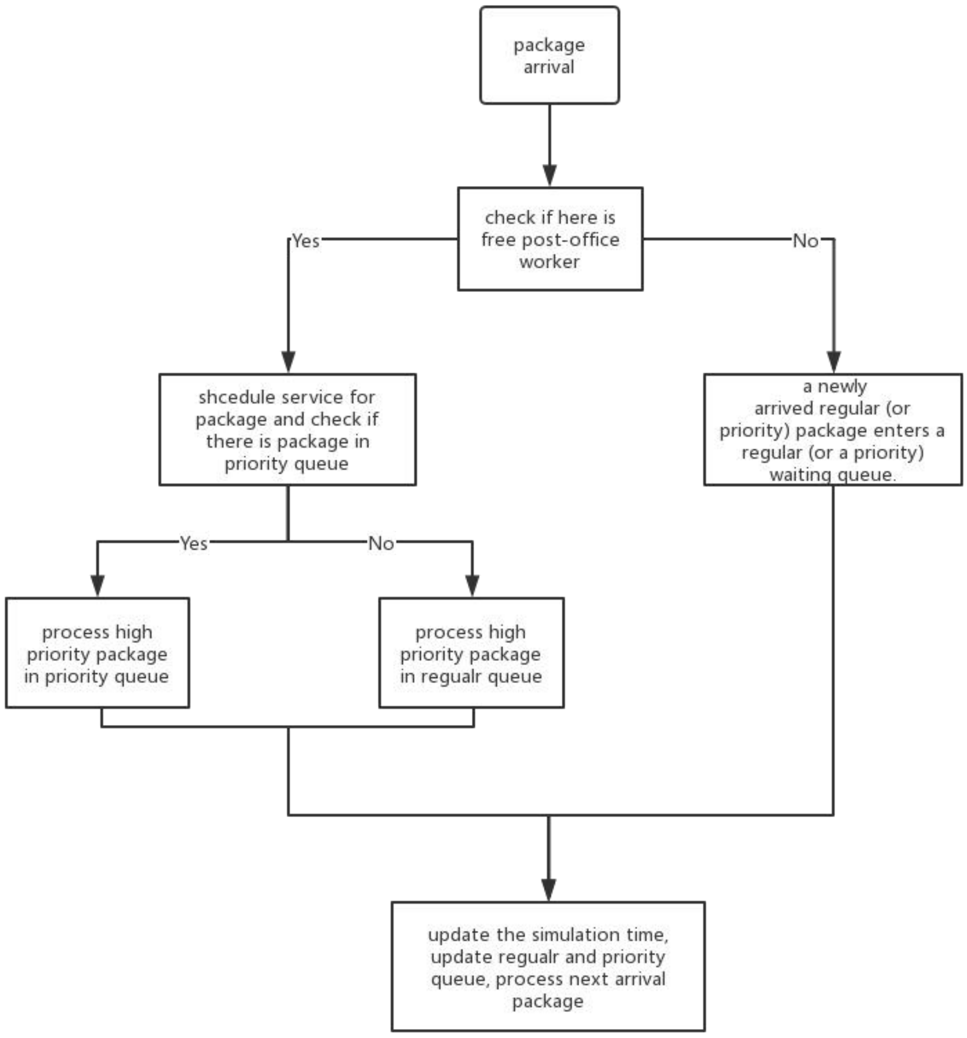
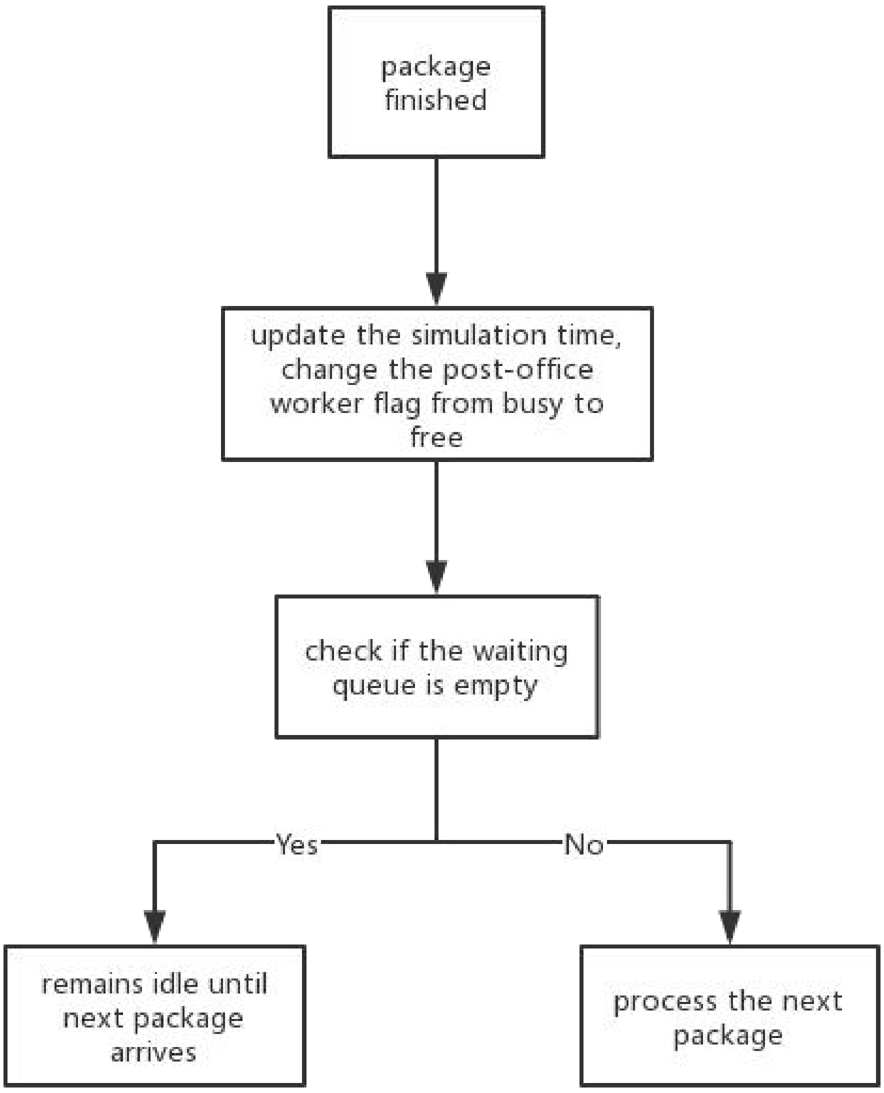

start
regular and priority packages, 9 post-office workers
a newly
arrived regular (or priority) package enters a regular (or a priority) waiting queue.
the
worker will fetch a package from a priority queue if there are waiting packages in both the regular and the priority queues
If all workers are busy with package processing, a newly
arrived regular (or priority) package enters a regular (or a priority) waiting queue.
As soon as a worker finishes a package processing, she either takes a new package
from the waiting queues or remains idle if there are no packages waiting.
Every event occurring in this system requires to change the state, as changing the number of packages in the correspond queue, and the time of the event needs to be recorded. The simulation will stop when t=T=2000.
continue
finish

Diagram 3: Sub-routines


package arrival
check if here is free post-office worker
Yes
No
shcedule service for package and check if there is package in priority queue
a newly arrived regular (or priority) package enters a regular (or a priority) waiting queue.
Yes
No
process high priority package in priority queue
process high priority package in regualr queue
update the simulation time, update regualr and priority queue, process next arrival package

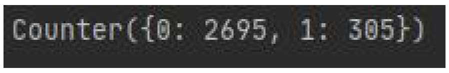
Sub-activity for package arrival event


package finished
update the simulation time, change the post-office worker flag from busy to free
check if the waiting queue is empty
Yes
No
remains idle until next package arrives
process the next package

Sub-activity for package finished event

(c) Results and Analysis. Using tables and figures, present a clear outcome of your study. Present the corresponding confidence intervals. [20 Marks]

At first, I set "maxNumber"representing max number of packages in the system to 3000, which means that there are a maximum of 3000 packages in this system. And then, I use Counter() class, an unordered collection where elements are stored as dictionary keys and their counts are stored as dictionary values, to count how many regular and priority packages in this system. The snapshot of result I listed below:



```
Counter({0: 2695, 1: 305})
```

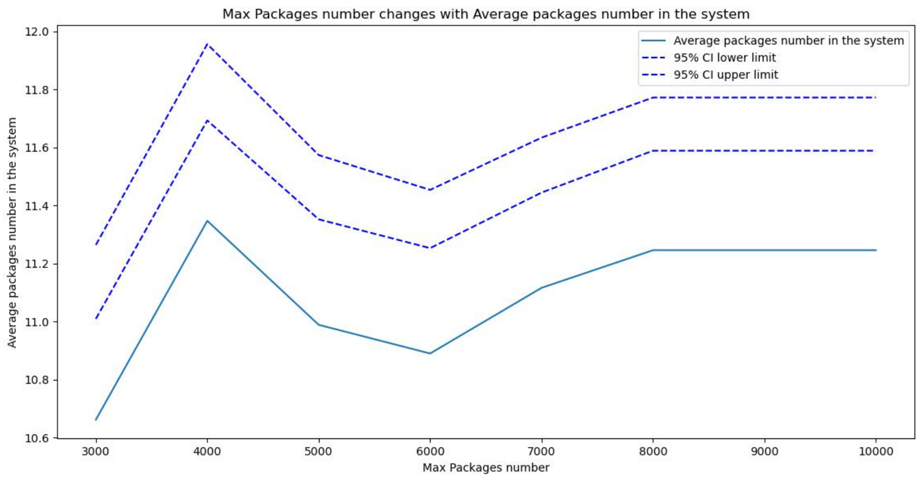
This means that there are 2695 regular packages and 305 priority packages. The proportion of regular packages in this system is 0.8983, which meets the criteria given by question: “a package is regular with probability 0.9”.

Moreover, we also get the result of the proportion of time that all workers are busy. The proportion is 0.69, which means that there will be about 0.69 proportion of time that all workers are busy in this time period.

Besides, I also get some other outcome which I make a table listed below:

Variables	**Value**	**95% confidence interval**
Average packages number in the system	10.6615	[11.0091,11.2638]
Average time in the system	2.7688	[2.6893,2.8484]
Actual average processing time	2.0223	[1.9500,2.0946]

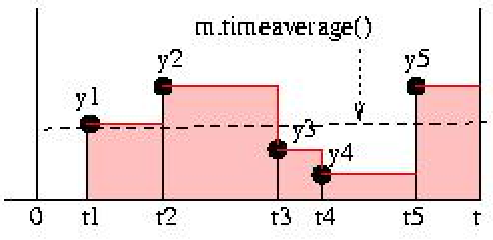
Because question doesn’t give limitation of package amount, I tuned “maxNumber” from 3000 to 10000. All figures of corresponding variables are listed below:



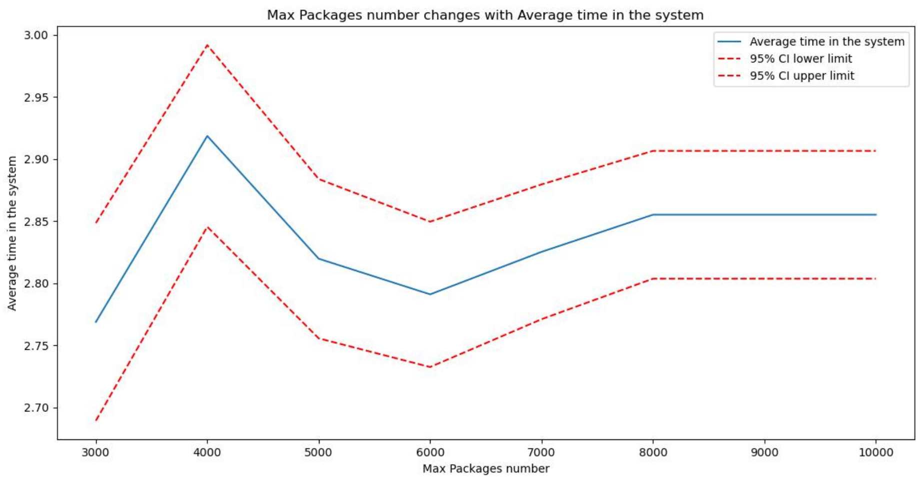
To calculate average packages number in the system, I use timeAverage([t]) function according to SimPy documentation rather than mean() function, because m.mean( ) function is the simple numerical average of the observed y values, ignoring the times at which they were made, which is m.total( )/m.count( ).

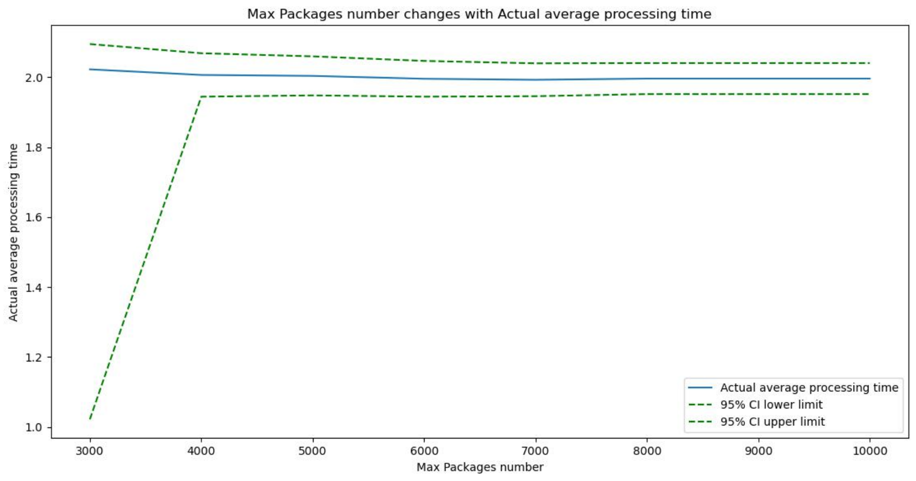
However, m.timeAverage([t]) calculates the time-weighted average of y, calculated from time 0 (or the last time m.reset([t]) was called) to time t (or to the current simulation time, now(), if t is missing).

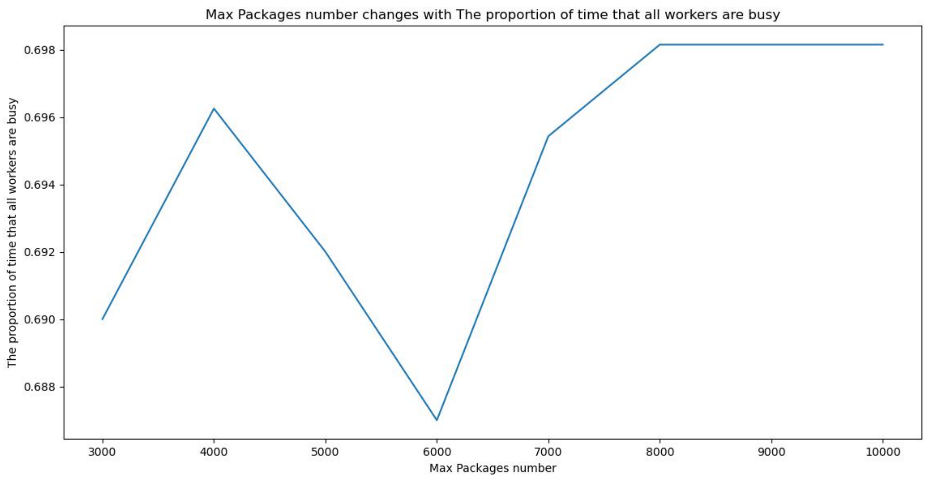
This is intended to measure the average of a quantity that always exists, such as the length of a queue or the amount in a Level. In discrete-event simulation such quantity changes are always instantaneous jumps occurring at events. The recorded times and new levels are sufficient information to calculate the average over time. The graph shown in the figure below illustrates the calculation. The total area under the line is calculated and divided by the total time of observation. For accurate time-average results y must be piecewise constant like this and observed just after each change in its value. That is, the y value observed must be the new value after the state change.



(The snapshot from https://pythonhosted.org/SimPy/Manuals/SManual.html#resources)

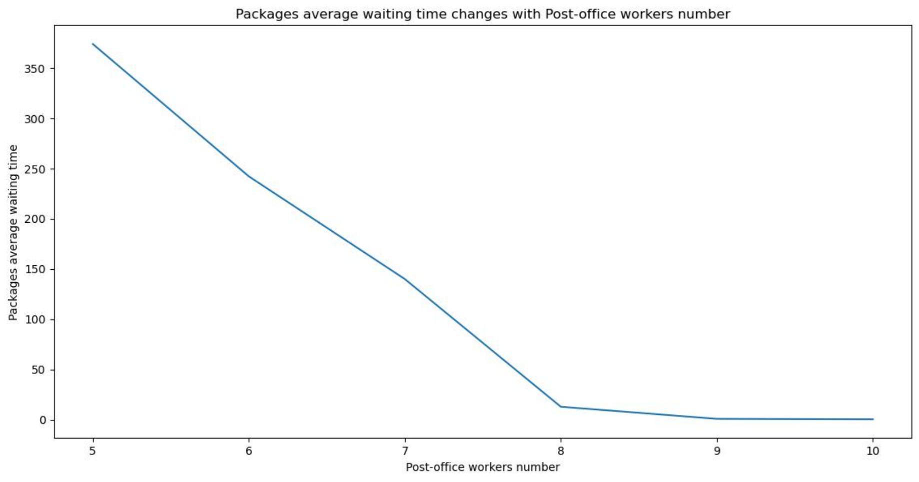


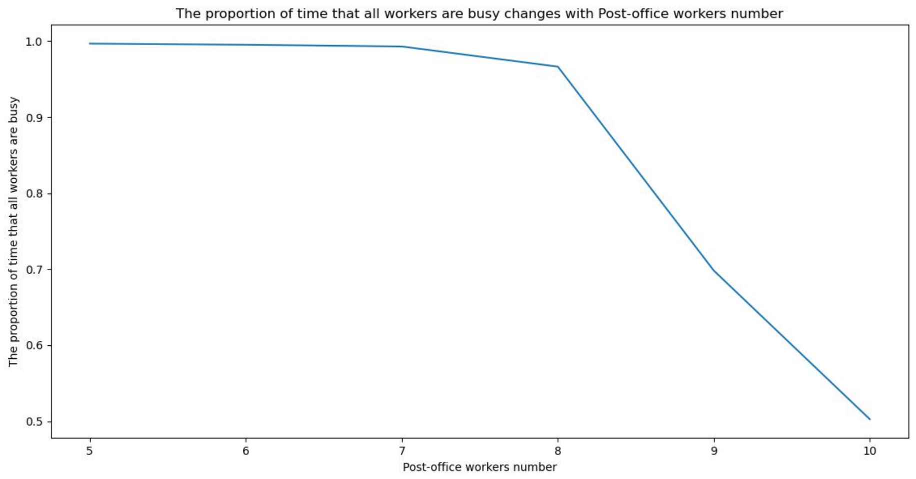




According to all plots I listed above, we could find that when max packages number from 8000 to 10000, the trend is very smooth, that is, a flat straight line. This means that starting with 8000 max packages number, these variable values will not change no matter how I increase max packages number. Therefore, we could find that the system has reach stability from a maximum of 8000 packages number.

Because we have known that the system has reach stability from 8000 max .packages number, I change the number of post-office workers base on "maxNumber"=1000 to check whether I need to increase or decrease workers number or not and change in the average waiting time of packages in the system. I list the plot of these two results below:





According to plots of post-office workers number changes with packages average waiting time and the proportion of time that all workers are busy, we could find that we should decrease the post-office number to 8 so that we could enhance workers' working efficiency. Because when "post-office number"=8, packages average waiting time is relatively low with high the proportion of time that all workers are busy.

(d) Formulate your conclusions. [20 Marks]
In this simulation, we have known that there are 9 workers in this post-office and regular and priority packages arrive to a post office according to a Poisson process with rate λ = 4 (4 packages per unit time on average) and the processing time is exponentially distributed with mean 2 (that is, 0.5 packages can be processed on average per unit time).

The purpose of this study is to find the proportion of time that all workers are busy during the day from t = 0 to T = 2000 and report the proportion along with 95% confidence interval. Hence, I develop a system to simulate the process of regular and priority packages and observe operational process of these packages throughout the system.

By this simulation system, we could find the average packages number and average time and actual average processing time in this system when I change max packages number from 3000 to 10000. To see the results more visually, I've listed a table below:

Max packages number	Average packages number	Average time	Actual average processing time	The proportion of time that all workers are busy	Priority packages number	Regular packages number
3000	10.6615	2.7688	2.0223	0.69	309	2691
4000	11.3471	2.9185	2.0063	0.69625	426	3574
5000	10.9885	2.8197	2.0036	0.692	523	4477
6000	10.8897	2.7910	1.9954	0.687	585	5415

7000	11.1160	2.8252	1.9925	0.6954285714285714	716	6284
8000	11.2457	2.8551	1.9959	0.6981467377506981	774	7104
9000	11.2457	2.8551	1.9959	0.6981467377506981	772	7106
10000	11.2457	2.8551	1.9959	0.6981467377506981	788	7090

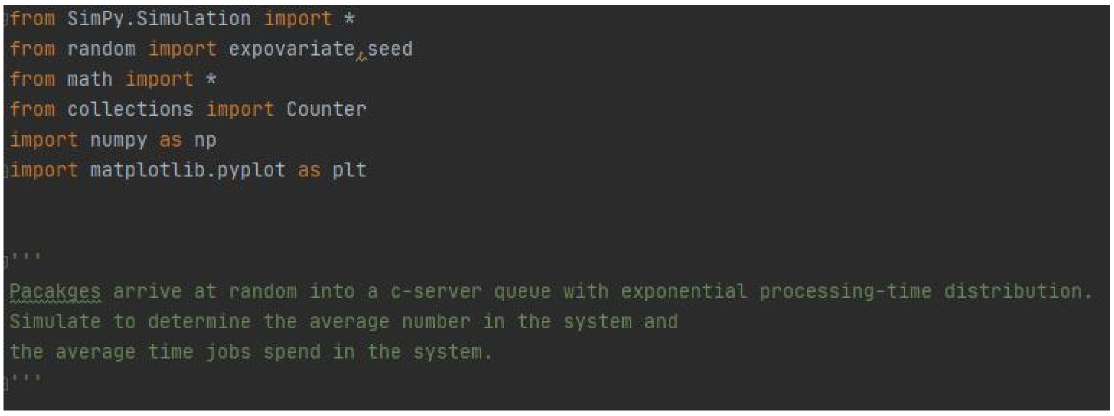
According to the table I listed above, we could find that when max packages number from 8000 to 10000, the trend is very smooth, that is, a flat straight line. This means that starting with 8000 max packages number, these variable values will not change no matter how I increase max packages number. Therefore, we could find that the system has reach stability from a maximum of 8000 packages number.

Because we have known that the system has reach stability from 8000 max .packages number, I change the number of post-office workers base on "maxNumber"=1000 to check whether I need to increase or decrease workers number or not and change in the average waiting time of packages in the system. I listed a table below:

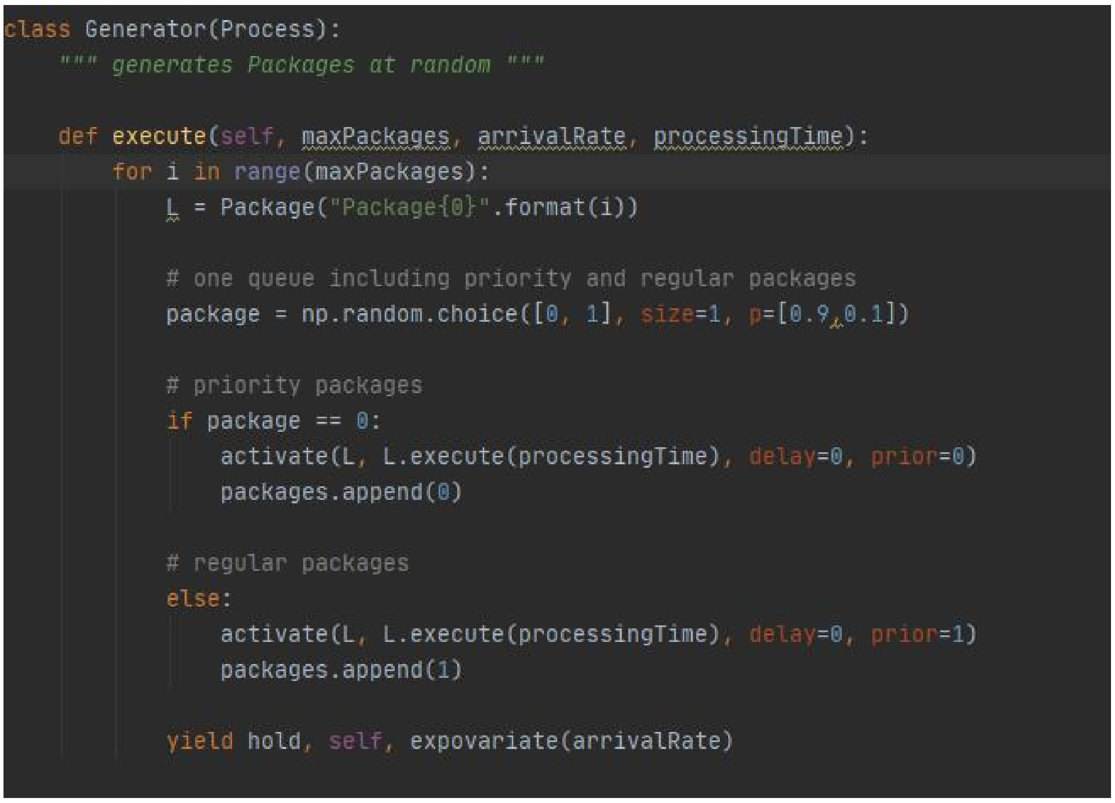
Workers number	**Average waiting time**	**All workers' busy proportion**
5	373.9899	0.9967832730197025
6	242.2265	0.9953043769914472
7	140.0121	0.9930131004366812
8	12.892999999999999	0.9665144596651446
9	0.8592000000000002	0.6981467377506981
10	0.4232	0.5026421741318571

We could know that the post office should decrease the post-office number to 8 so that it could enhance workers' working efficiency. Because when "post-office number"=8, packages average waiting time is relatively low with high the proportion of time that all workers are busy.

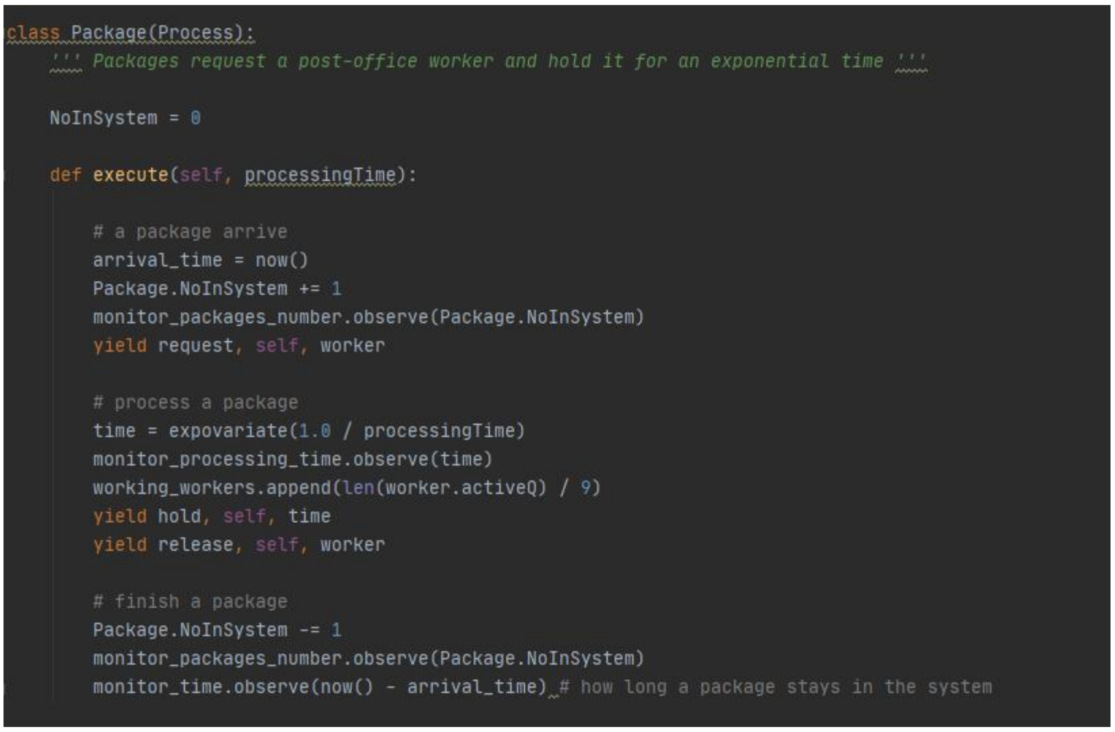
(e) Appendix. Include all code files used. Explain their interaction and provide a clear and well-commented code. [10 Marks]



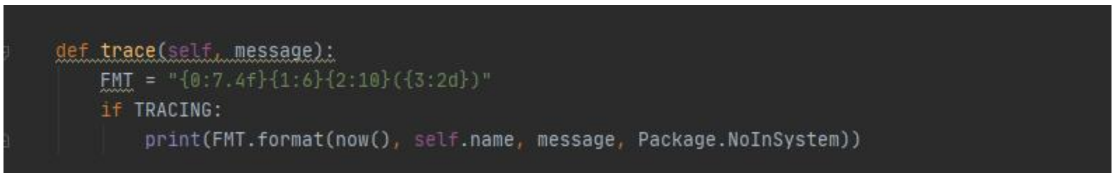
```
from SimPy.Simulation import *
from random import expovariate,seed
from math import *
from collections import Counter
import numpy as np
import matplotlib.pyplot as plt


'''
Pacakges arrive at random into a c-server queue with exponential processing-time distribution.
Simulate to determine the average number in the system and
the average time jobs spend in the system.
'''
```



```python
class Generator(Process):
    """ generates Packages at random """

    def execute(self, maxPackages, arrivalRate, processingTime):
        for i in range(maxPackages):
            L = Package("Package{0}".format(i))

            # one queue including priority and regular packages
            package = np.random.choice([0, 1], size=1, p=[0.9,0.1])

            # priority packages
            if package == 0:
                activate(L, L.execute(processingTime), delay=0, prior=0)
                packages.append(0)

            # regular packages
            else:
                activate(L, L.execute(processingTime), delay=0, prior=1)
                packages.append(1)

            yield hold, self, expovariate(arrivalRate)
```



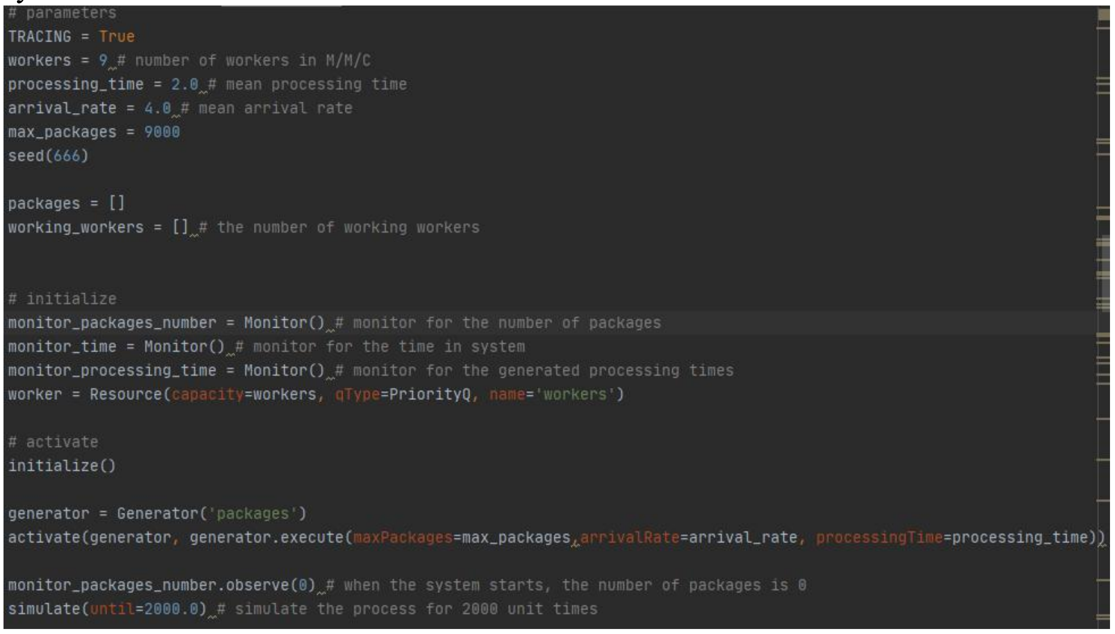
```python
class Package(Process):
    ''' Packages request a post-office worker and hold it for an exponential time '''

    NoInSystem = 0

    def execute(self, processingTime):

        # a package arrive
        arrival_time = now()
        Package.NoInSystem += 1
        monitor_packages_number.observe(Package.NoInSystem)
        yield request, self, worker

        # process a package
        time = expovariate(1.0 / processingTime)
        monitor_processing_time.observe(time)
        working_workers.append(len(worker.activeQ) / 9)
        yield hold, self, time
        yield release, self, worker

        # finish a package
        Package.NoInSystem -= 1
        monitor_packages_number.observe(Package.NoInSystem)
        monitor_time.observe(now() - arrival_time) # how long a package stays in the system
```



```python
    def trace(self, message):
        FMT = "{0:7.4f}{1:6}{2:10}({3:2d})"
        if TRACING:
            print(FMT.format(now(), self.name, message, Package.NoInSystem))
```

Firstly, I create a class called “Generator”, which acts as a packages generator to generate package. Then, I use the built-in function of this class.

Next, I create a class called “Package”, which aims to monitor the whole discrete system.

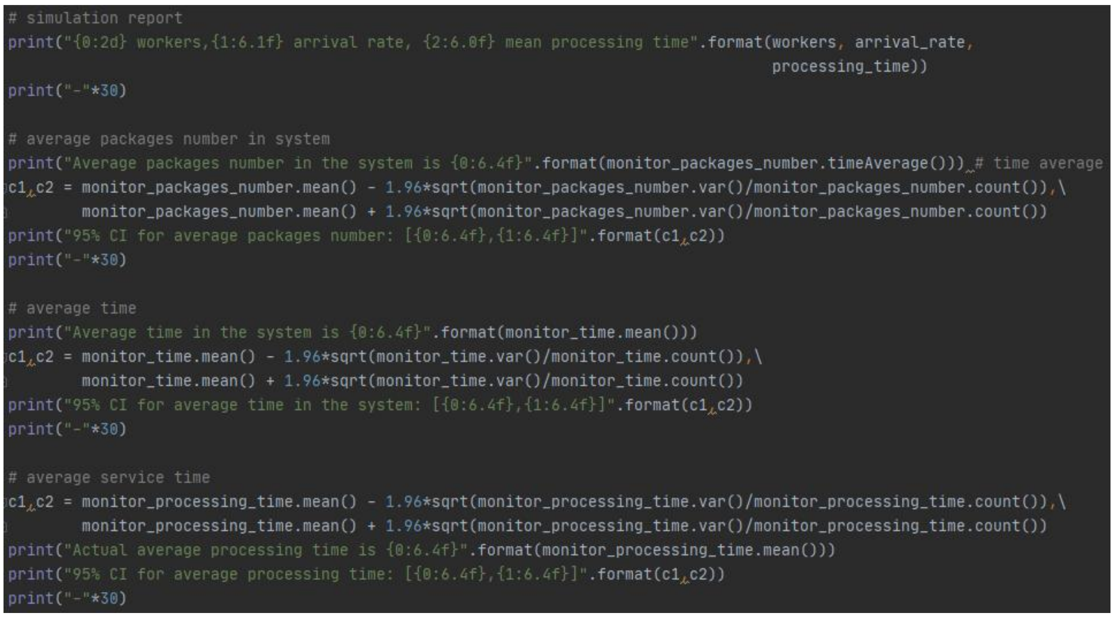


```python
# parameters
TRACING = True
workers = 9 # number of workers in M/M/C
processing_time = 2.0 # mean processing time
arrival_rate = 4.0 # mean arrival rate
max_packages = 9000
seed(666)

packages = []
working_workers = [] # the number of working workers


# initialize
monitor_packages_number = Monitor() # monitor for the number of packages
monitor_time = Monitor() # monitor for the time in system
monitor_processing_time = Monitor() # monitor for the generated processing times
worker = Resource(capacity=workers, qType=PriorityQ, name='workers')

# activate
initialize()

generator = Generator('packages')
activate(generator, generator.execute(maxPackages=max_packages,arrivalRate=arrival_rate, processingTime=processing_time))

monitor_packages_number.observe(0) # when the system starts, the number of packages is 0
simulate(until=2000.0) # simulate the process for 2000 unit times
```

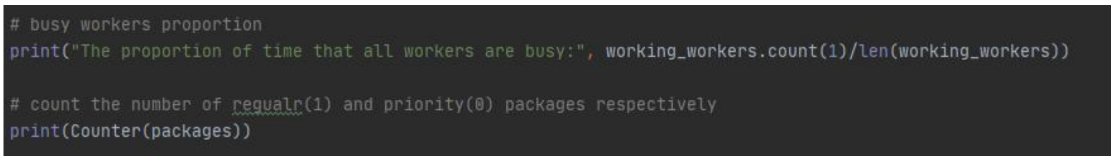
Because I want to get packages number, the time of the package in the system and the processing time, I use Monitor() class to monitor the operational process of packages and records the corresponding results.

And then, I use Resource object, where capacity=workers representing the total number of workers in the system, qType=PriorityQ representing there is priority queue in the system.

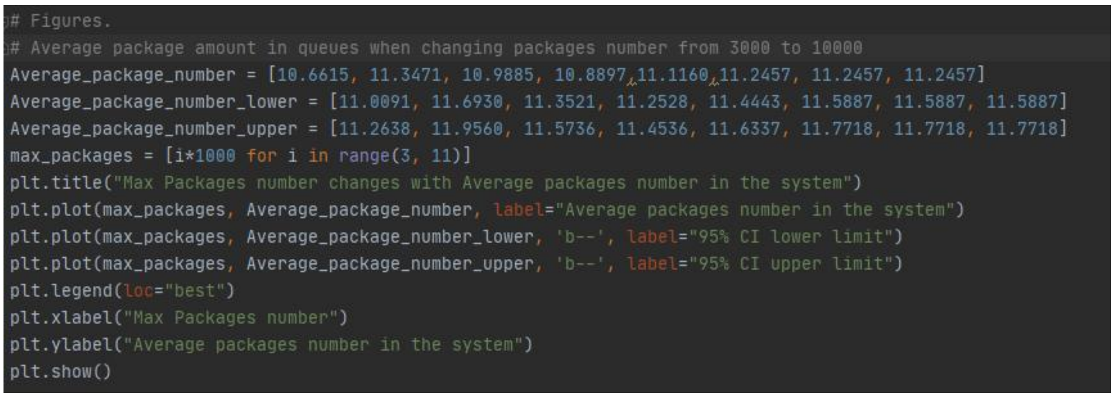
Next, I use initialize() function to initialize the simulation and activate() fucntion to activate simulation process.



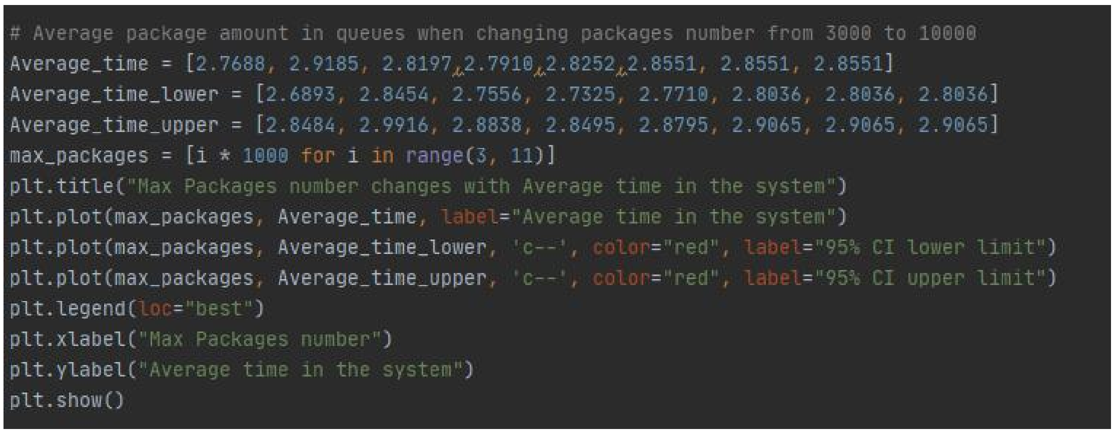
```python
# simulation report
print("{0:2d} workers,{1:6.1f} arrival rate, {2:6.0f} mean processing time".format(workers, arrival_rate,
                                                                                 processing_time))
print("-"*30)

# average packages number in system
print("Average packages number in the system is {0:6.4f}".format(monitor_packages_number.timeAverage())) # time average
c1,c2 = monitor_packages_number.mean() - 1.96*sqrt(monitor_packages_number.var()/monitor_packages_number.count()),\
        monitor_packages_number.mean() + 1.96*sqrt(monitor_packages_number.var()/monitor_packages_number.count())
print("95% CI for average packages number: [{0:6.4f},{1:6.4f}]".format(c1,c2))
print("-"*30)

# average time
print("Average time in the system is {0:6.4f}".format(monitor_time.mean()))
c1,c2 = monitor_time.mean() - 1.96*sqrt(monitor_time.var()/monitor_time.count()),\
        monitor_time.mean() + 1.96*sqrt(monitor_time.var()/monitor_time.count())
print("95% CI for average time in the system: [{0:6.4f},{1:6.4f}]".format(c1,c2))
print("-"*30)

# average service time
c1,c2 = monitor_processing_time.mean() - 1.96*sqrt(monitor_processing_time.var()/monitor_processing_time.count()),\
        monitor_processing_time.mean() + 1.96*sqrt(monitor_processing_time.var()/monitor_processing_time.count())
print("Actual average processing time is {0:6.4f}".format(monitor_processing_time.mean()))
print("95% CI for average processing time: [{0:6.4f},{1:6.4f}]".format(c1,c2))
print("-"*30)
```



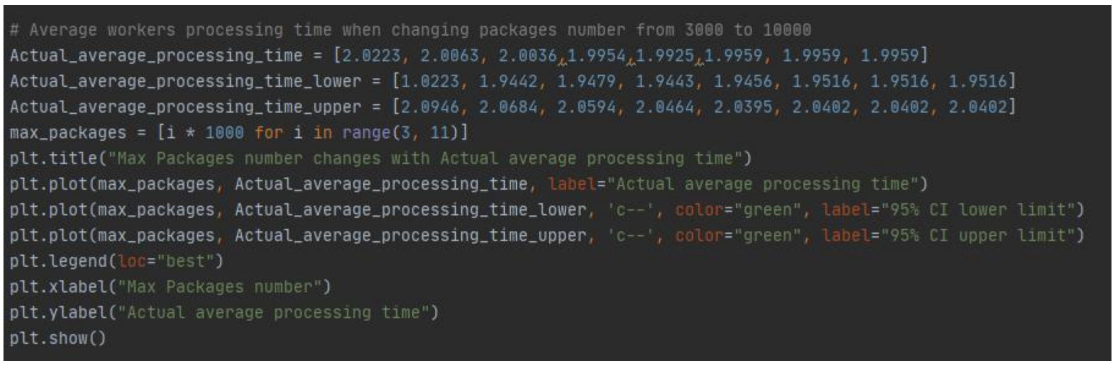
```python
# busy workers proportion
print("The proportion of time that all workers are busy:", working_workers.count(1)/len(working_workers))

# count the number of regualr(1) and priority(0) packages respectively
print(Counter(packages))
```



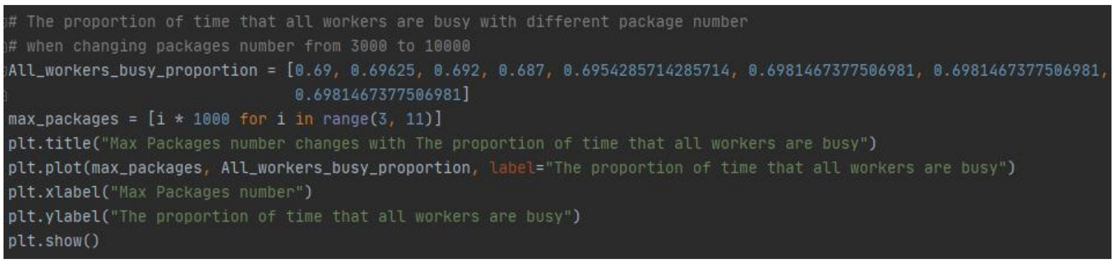
```python
# Figures.
# Average package amount in queues when changing packages number from 3000 to 10000
Average_package_number = [10.6615, 11.3471, 10.9885, 10.8897,11.1160,11.2457, 11.2457, 11.2457]
Average_package_number_lower = [11.0091, 11.6930, 11.3521, 11.2528, 11.4443, 11.5887, 11.5887, 11.5887]
Average_package_number_upper = [11.2638, 11.9560, 11.5736, 11.4536, 11.6337, 11.7718, 11.7718, 11.7718]
max_packages = [i*1000 for i in range(3, 11)]
plt.title("Max Packages number changes with Average packages number in the system")
plt.plot(max_packages, Average_package_number, label="Average packages number in the system")
plt.plot(max_packages, Average_package_number_lower, 'b--', label="95% CI lower limit")
plt.plot(max_packages, Average_package_number_upper, 'b--', label="95% CI upper limit")
plt.legend(loc="best")
plt.xlabel("Max Packages number")
plt.ylabel("Average packages number in the system")
plt.show()
```



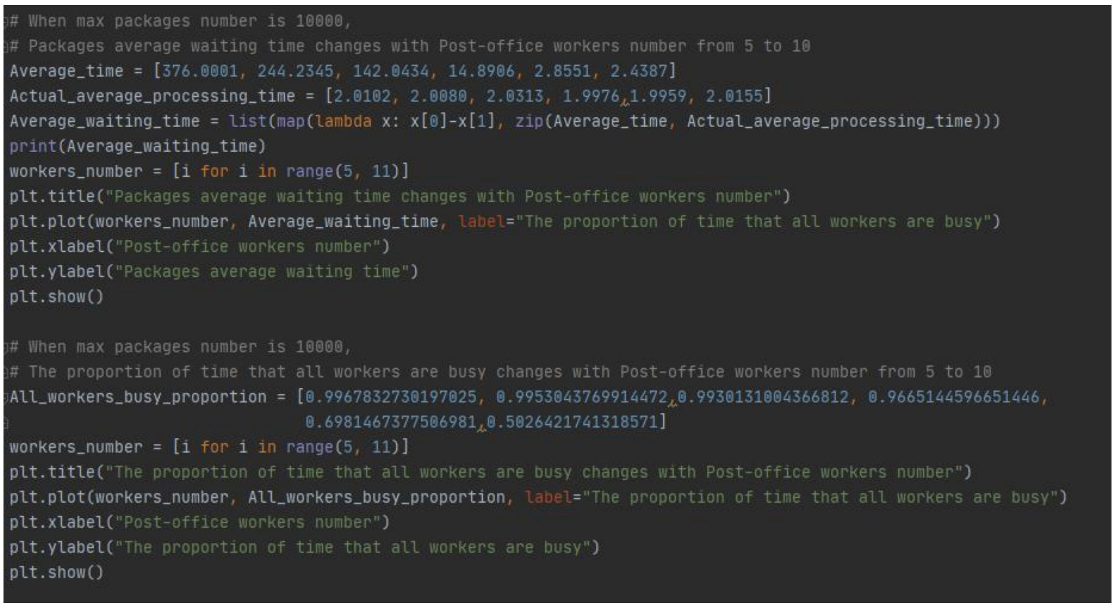
```python
# Average package amount in queues when changing packages number from 3000 to 10000
Average_time = [2.7688, 2.9185, 2.8197,2.7910,2.8252,2.8551, 2.8551, 2.8551]
Average_time_lower = [2.6893, 2.8454, 2.7556, 2.7325, 2.7710, 2.8036, 2.8036, 2.8036]
Average_time_upper = [2.8484, 2.9916, 2.8838, 2.8495, 2.8795, 2.9065, 2.9065, 2.9065]
max_packages = [i * 1000 for i in range(3, 11)]
plt.title("Max Packages number changes with Average time in the system")
plt.plot(max_packages, Average_time, label="Average time in the system")
plt.plot(max_packages, Average_time_lower, 'c--', color="red", label="95% CI lower limit")
plt.plot(max_packages, Average_time_upper, 'c--', color="red", label="95% CI upper limit")
plt.legend(loc="best")
plt.xlabel("Max Packages number")
plt.ylabel("Average time in the system")
plt.show()
```



```python
# Average workers processing time when changing packages number from 3000 to 10000
Actual_average_processing_time = [2.0223, 2.0063, 2.0036,1.9954,1.9925,1.9959, 1.9959, 1.9959]
Actual_average_processing_time_lower = [1.0223, 1.9442, 1.9479, 1.9443, 1.9456, 1.9516, 1.9516, 1.9516]
Actual_average_processing_time_upper = [2.0946, 2.0684, 2.0594, 2.0464, 2.0395, 2.0402, 2.0402, 2.0402]
max_packages = [i * 1000 for i in range(3, 11)]
plt.title("Max Packages number changes with Actual average processing time")
plt.plot(max_packages, Actual_average_processing_time, label="Actual average processing time")
plt.plot(max_packages, Actual_average_processing_time_lower, 'c--', color="green", label="95% CI lower limit")
plt.plot(max_packages, Actual_average_processing_time_upper, 'c--', color="green", label="95% CI upper limit")
plt.legend(loc="best")
plt.xlabel("Max Packages number")
plt.ylabel("Actual average processing time")
plt.show()
```



```python
# The proportion of time that all workers are busy with different package number
# when changing packages number from 3000 to 10000
All_workers_busy_proportion = [0.69, 0.69625, 0.692, 0.687, 0.6954285714285714, 0.6981467377506981, 0.6981467377506981,
                               0.6981467377506981]
max_packages = [i * 1000 for i in range(3, 11)]
plt.title("Max Packages number changes with The proportion of time that all workers are busy")
plt.plot(max_packages, All_workers_busy_proportion, label="The proportion of time that all workers are busy")
plt.xlabel("Max Packages number")
plt.ylabel("The proportion of time that all workers are busy")
plt.show()
```



```python
# When max packages number is 10000,
# Packages average waiting time changes with Post-office workers number from 5 to 10
Average_time = [376.0001, 244.2345, 142.0434, 14.8906, 2.8551, 2.4387]
Actual_average_processing_time = [2.0102, 2.0080, 2.0313, 1.9976,1.9959, 2.0155]
Average_waiting_time = list(map(lambda x: x[0]-x[1], zip(Average_time, Actual_average_processing_time)))
print(Average_waiting_time)
workers_number = [i for i in range(5, 11)]
plt.title("Packages average waiting time changes with Post-office workers number")
plt.plot(workers_number, Average_waiting_time, label="The proportion of time that all workers are busy")
plt.xlabel("Post-office workers number")
plt.ylabel("Packages average waiting time")
plt.show()

# When max packages number is 10000,
# The proportion of time that all workers are busy changes with Post-office workers number from 5 to 10
All_workers_busy_proportion = [0.9967832730197025, 0.9953043769914472,0.9930131004366812, 0.9665144596651446,
                               0.6981467377506981,0.5026421741318571]
workers_number = [i for i in range(5, 11)]
plt.title("The proportion of time that all workers are busy changes with Post-office workers number")
plt.plot(workers_number, All_workers_busy_proportion, label="The proportion of time that all workers are busy")
plt.xlabel("Post-office workers number")
plt.ylabel("The proportion of time that all workers are busy")
plt.show()
```

Reference

Müller, Klaus, Tony Vignaux, Ontje Lünsdorf, and Stefan Scherfke. "SimPy Classic Manual." SimPy Documentation Release 2.3.4, February 24, 2018.(Page 110 to 114) https://readthedocs.org/projects/simpyclassic/downloads/pdf/latest/.

"SimPy For First Time Users." SimPy For First Time Users - SimPy v2.2 documentation. Accessed November 6, 2020. https://pythonhosted.org/SimPy/Manuals/SManual.html.