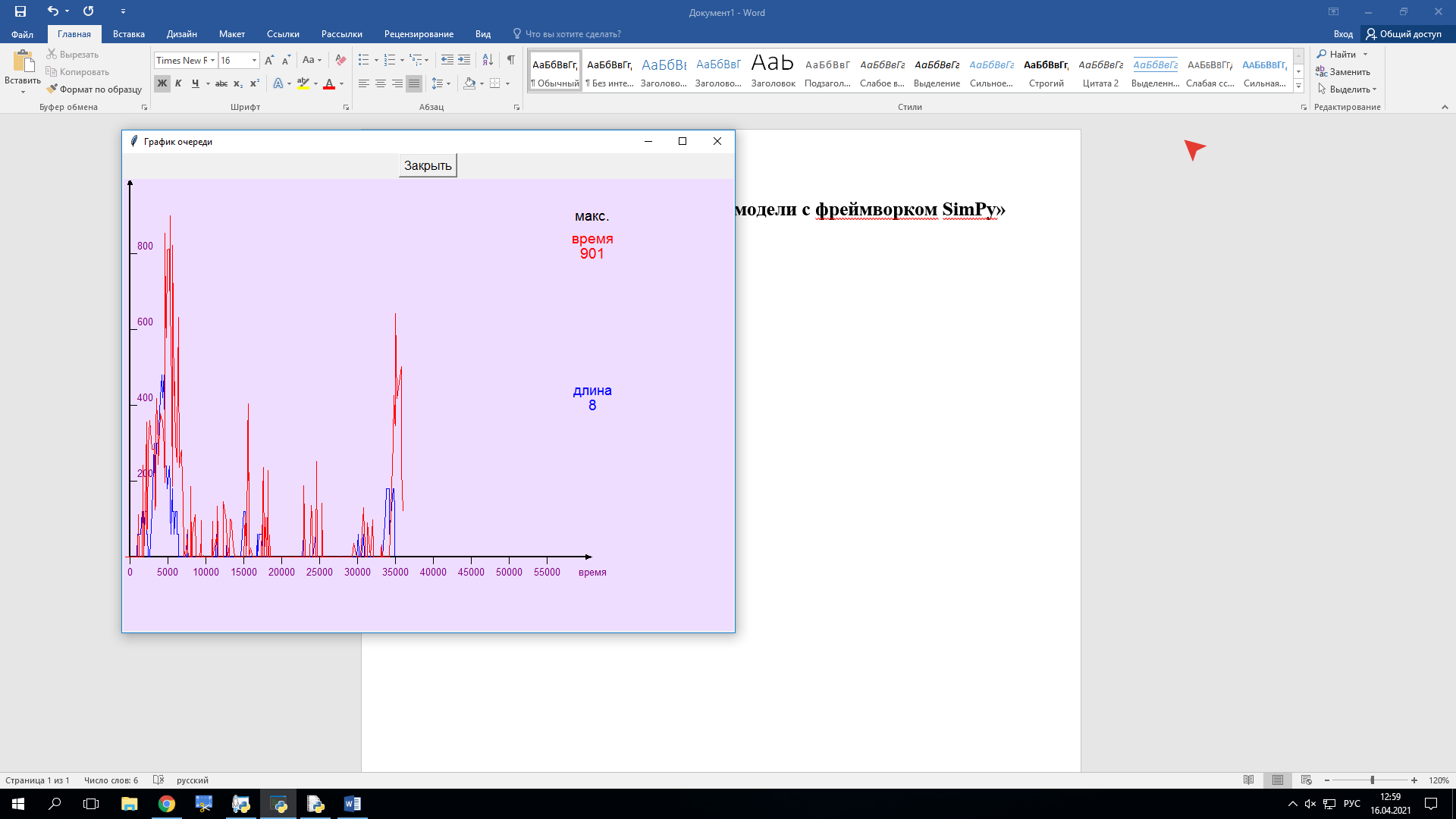
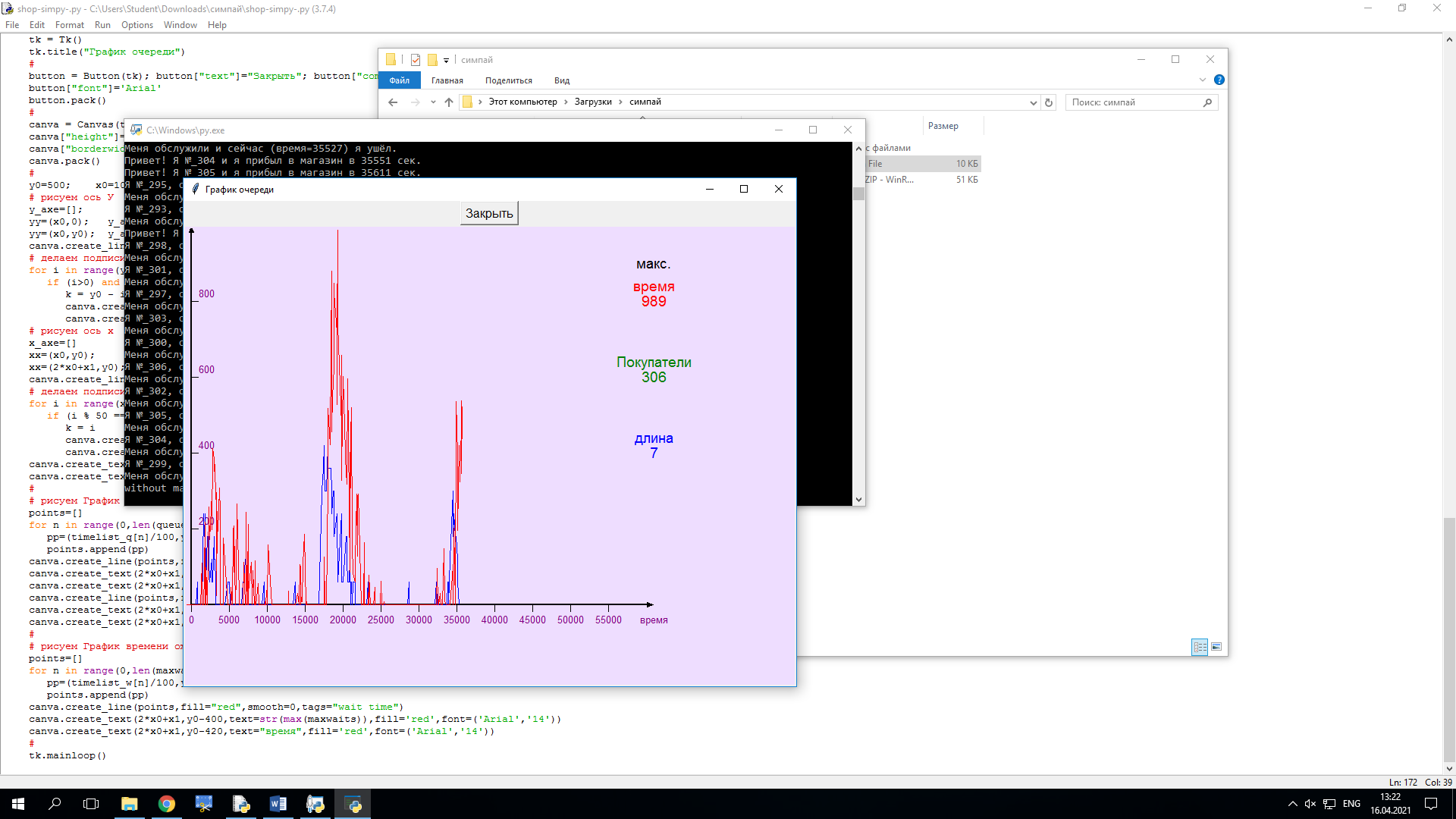
**«Настройка имитационной модели с фреймворком SimPy»**

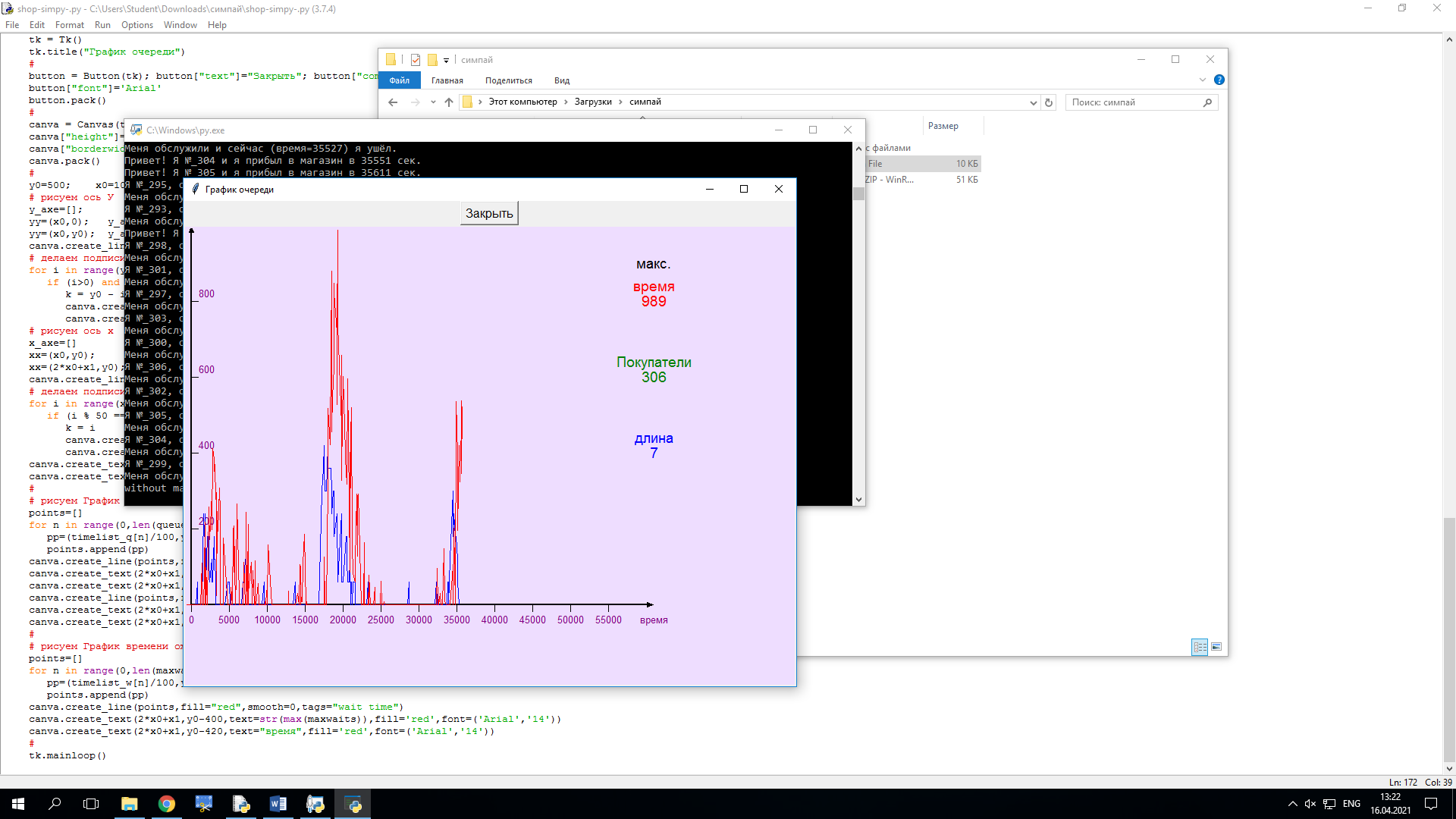
Изначальный график (**config1**)

****

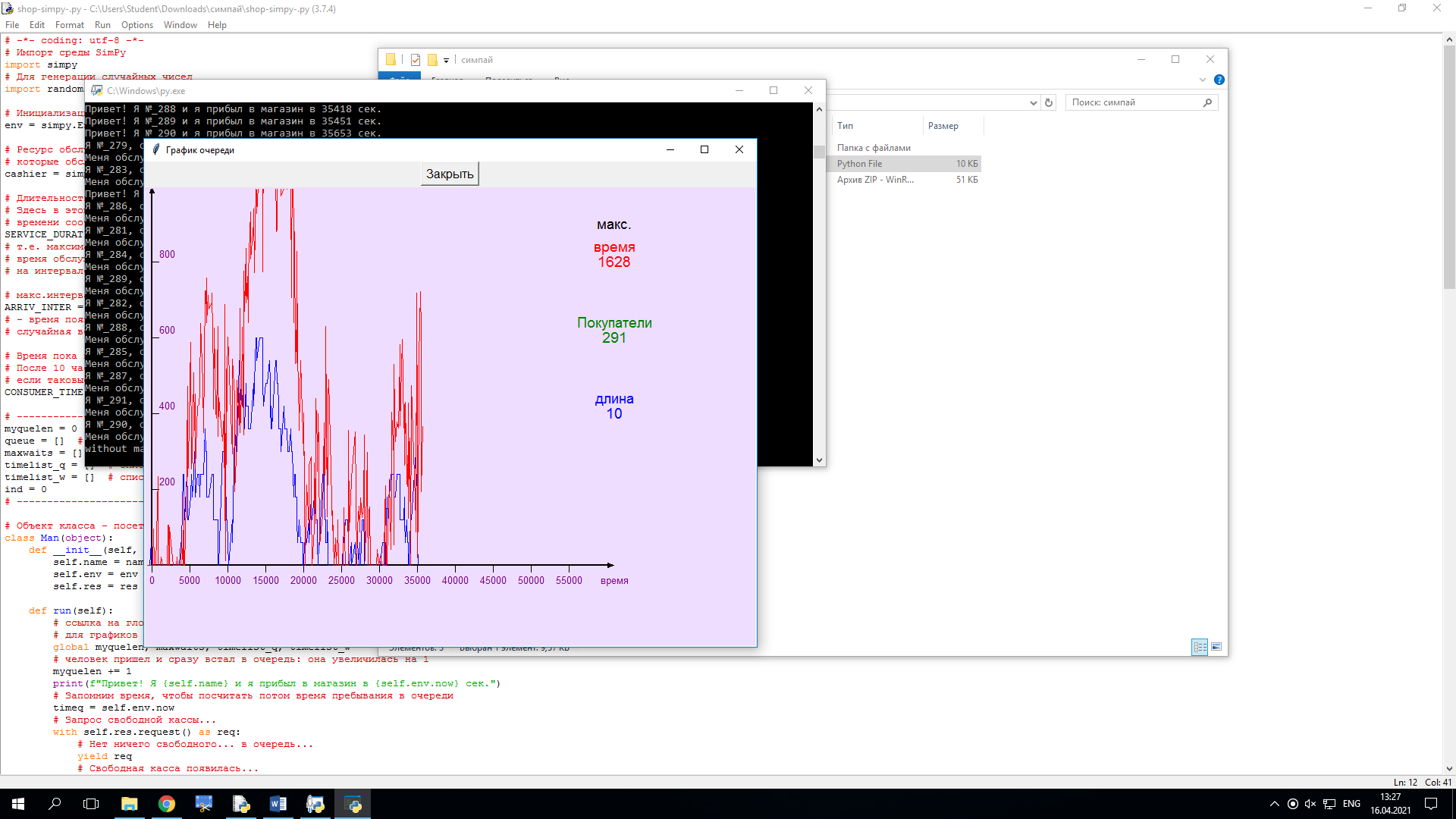
Вывели количество покупателей

****

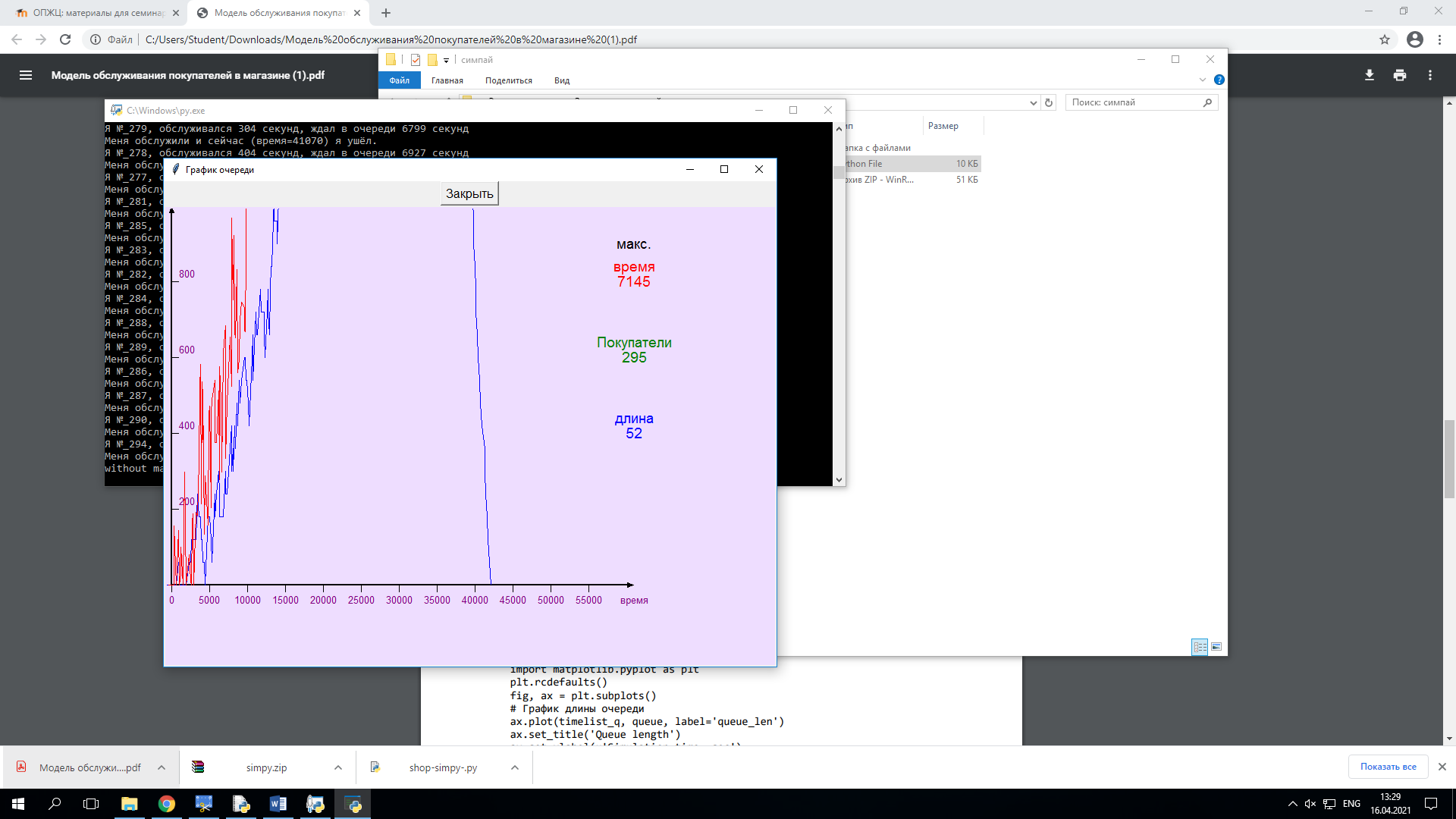
**Config 1**

****

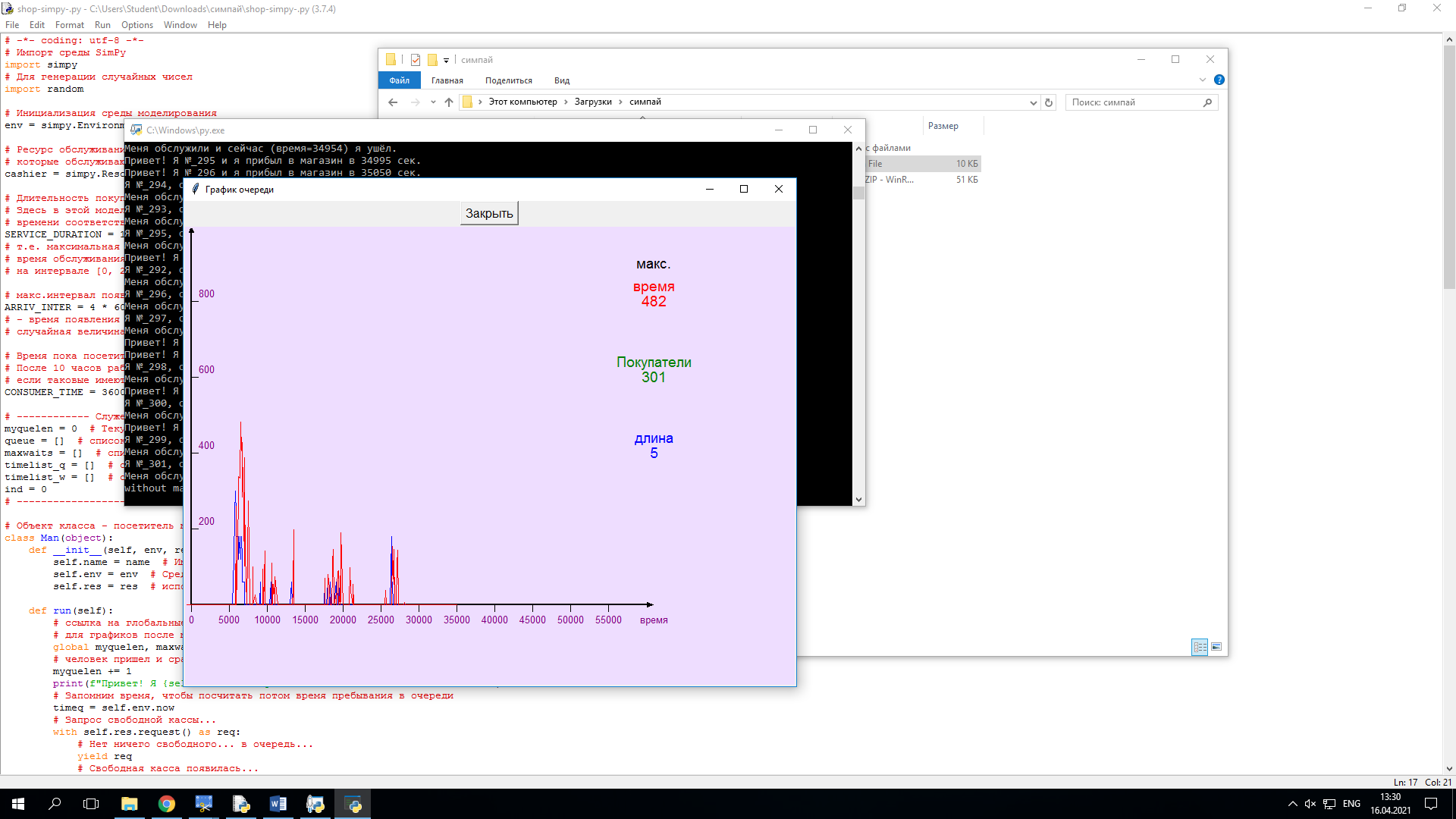
**Config 2**

****

**Config 3**

****

**Config 4**



**Код:**# -\*- coding: utf-8 -\*-

# Импорт среды SimPy

import simpy

# Для генерации случайных чисел

import random

# Инициализация среды моделирования

env = simpy.Environment()

# Ресурс обслуживания; в данном случае - capacity - число касс,

# которые обслуживают покупателей

cashier = simpy.Resource(env, capacity=4)

# Длительность покупки\получения услуги

# Здесь в этой модели мы полагаем, что единица модельного

# времени соответствует 1 секунде реального времени

SERVICE\_DURATION = 10 \* 60

# т.е. максимальная длительность обслуживания 20 мин

# время обслуживания равномерно распределенная случайная величина

# на интервале [0, 20 \* 60]

# макс.интервал появления нового посетителя магазина

ARRIV\_INTER = 4 \* 60

# - время появления нового посетителя равномерно распределенная

# случайная величина на интервале [0, 4 \* 60]

# Время пока посетителей пускают в магазин (10 часов)

# После 10 часов работы магазин закрывают и обслуживают только оставшихся,

# если таковые имеются...

CONSUMER\_TIME = 3600 \* 10

# ------------ Служебные параметры для статистики --------

myquelen = 0 # Текущая длина очереди

queue = [] # список очереди для построения графика

maxwaits = [] # список времен ожидания в очереди для построения графика

timelist\_q = [] # список времен, соответствующий длине очереди queue

timelist\_w = [] # список времен, соответствующий временам ожидания

ind = 0

# -----------------------------------------------------------------

# Объект класса - посетитель магазина

class Man(object):

def \_\_init\_\_(self, env, res, name='default'):

self.name = name # Имя посетителя, чтобы их различать

self.env = env # Среда моделирования

self.res = res # используемый при моделировании ресурс,- касса

def run(self):

# ссылка на глобальные счетчики статистики

# для графиков после моделирования

global myquelen, maxwaits, timelist\_q, timelist\_w

# человек пришел и сразу встал в очередь: она увеличилась на 1

myquelen += 1

print(f"Привет! Я {self.name} и я прибыл в магазин в {self.env.now} сек.")

# Запомним время, чтобы посчитать потом время пребывания в очереди

timeq = self.env.now

# Запрос свободной кассы...

with self.res.request() as req:

# Нет ничего свободного... в очередь...

yield req

# Свободная касса появилась...

# Человек поступает на обслуживание и очередь уменьшается на 1

myquelen -= 1

# запомним текущую длину очереди

queue.append(myquelen)

# запомним текущее время события

timelist\_q.append(self.env.now)

# вспомогательная переменная (время, проведенное в очереди = wait - timeq)

wait = self.env.now

# время обслуживания - случайное число, генерируем его

serving\_duration = random.randint(0, SERVICE\_DURATION)

# обслуживаемся в кассе...

yield self.env.timeout(serving\_duration)

# Обслужили

print(f"Я {self.name}, обслуживался {serving\_duration} секунд, ждал в очереди {wait-timeq} секунд")

# Запомним время проведенное в очереди

maxwaits.append(wait-timeq)

# Запомним текущее время события

timelist\_w.append(self.env.now)

print(f"Меня обслужили и сейчас (время={self.env.now}) я ушёл.")

# Источник посетителей предполагает, что посетители приходят

# 10 часов от начала работы магазина, далее поступление новых прекращается

def source\_men(env):

global ind

while env.now < (CONSUMER\_TIME - ARRIV\_INTER): # Посетители приходят 10 часов

ind += 1

yield env.timeout(random.randint(0, ARRIV\_INTER))

man = Man(env, cashier, name='№\_%s' % ind)

env.process(man.run())

# Добавляем процесс появления в магазине новых посетителей

env.process(source\_men(env))

# Запускаем процесс моделирования, полагая, что

# один шаг моделирования - 1 секунда реального времени;

# процесс моделирования составляет 12 часов;

# посетители входят только 10 часов,

# далее обслуживаются оставшиеся в очереди, если таковые имеются

env.run(until=12 \* 60 \* 60)

# ------------------------ Выводим результаты моделирования в виде графиков

# Должен быть установлен пакет matplotlib,

# если нет, то используем canvas

try:

import matplotlib.pyplot as plt

plt.rcdefaults()

fig, ax = plt.subplots() #figure()

# График длины очереди

ax.plot(timelist\_q, queue, label='queue\_len')

ax.set\_title('Queue length')

ax.set\_xlabel(u'Simulation time, sec')

ax.set\_ylabel(u'Current queue length, #')

fig, bx = plt.subplots() #figure()

# График времени ожидания в очереди

bx.plot(timelist\_w, maxwaits)

bx.set\_title(u'Waiting time')

bx.set\_xlabel(u'Simulation time, sec')

bx.set\_ylabel(u'Somebody waits..., sec')

plt.show()

# --------------------------------------------------------------------------------------

except ImportError:

print('without matplotlib - use tkinter')

from tkinter import \*

#

tk = Tk()

tk.title("График очереди")

#

button = Button(tk); button["text"]="Закрыть"; button["command"]= tk.destroy

button["font"]='Arial'

button.pack()

#

canva = Canvas(tk)

canva["height"]=600; canva["width"]=800; canva["background"]="#eeddff"

canva["borderwidth"]=2

canva.pack()

#

y0=500; x0=10; x1=600; dx=1

# рисуем ось У

y\_axe=[];

yy=(x0,0); y\_axe.append(yy)

yy=(x0,y0); y\_axe.append(yy)

canva.create\_line(y\_axe,fill='black',width=2,arrow=FIRST)

# делаем подписи на оси

for i in range(y0):

if (i>0) and (i%100==0):

k = y0 - i

canva.create\_line(x0, k, x0+10, k, width = 0.5, fill = 'black')

canva.create\_text(x0+20, k-10,text=str(i\*2),fill='purple',font=('Arial','10'))

# рисуем ось х

x\_axe=[]

xx=(x0,y0); x\_axe.append(xx)

xx=(2\*x0+x1,y0); x\_axe.append(xx)

canva.create\_line(x\_axe,fill='black',width=2,arrow=LAST)

# делаем подписи на оси

for i in range(x1):

if (i % 50 == 0):

k = i

canva.create\_line(x0+i, y0, x0+i, y0+10, width = 0.5, fill = 'black')

canva.create\_text(x0+i, y0+20, text=str(i\*100),fill='purple',font=('Arial','10'))

canva.create\_text(2\*x0+x1,y0+20,text="время",fill='purple', font=('Arial','10'))

canva.create\_text(2\*x0+x1,50,text='макс.',fill='black', font=('Arial','14'))

#

# рисуем График длины очереди

points=[]

for n in range(0,len(queue)):

pp=(timelist\_q[n]/100,y0-queue[n]\*30)

points.append(pp)

canva.create\_line(points,fill="blue",smooth=0,tags="queue count")

canva.create\_text(2\*x0+x1,y0-200,text=str(max(queue)),fill='blue',font=('Arial','14'))

canva.create\_text(2\*x0+x1,y0-220,text="длина",fill='blue',font=('Arial','14'))

canva.create\_line(points,fill="blue",smooth=0,tags="queue count")

canva.create\_text(2\*x0+x1,y0-300,text=str(ind),fill='green',font=('Arial','14'))

canva.create\_text(2\*x0+x1,y0-320,text="Покупатели",fill='green',font=('Arial','14'))

#

# рисуем График времени ожидания в очереди

points=[]

for n in range(0,len(maxwaits)):

pp=(timelist\_w[n]/100,y0-maxwaits[n]/2)

points.append(pp)

canva.create\_line(points,fill="red",smooth=0,tags="wait time")

canva.create\_text(2\*x0+x1,y0-400,text=str(max(maxwaits)),fill='red',font=('Arial','14'))

canva.create\_text(2\*x0+x1,y0-420,text="время",fill='red',font=('Arial','14'))

#

tk.mainloop()