

Практическая №3

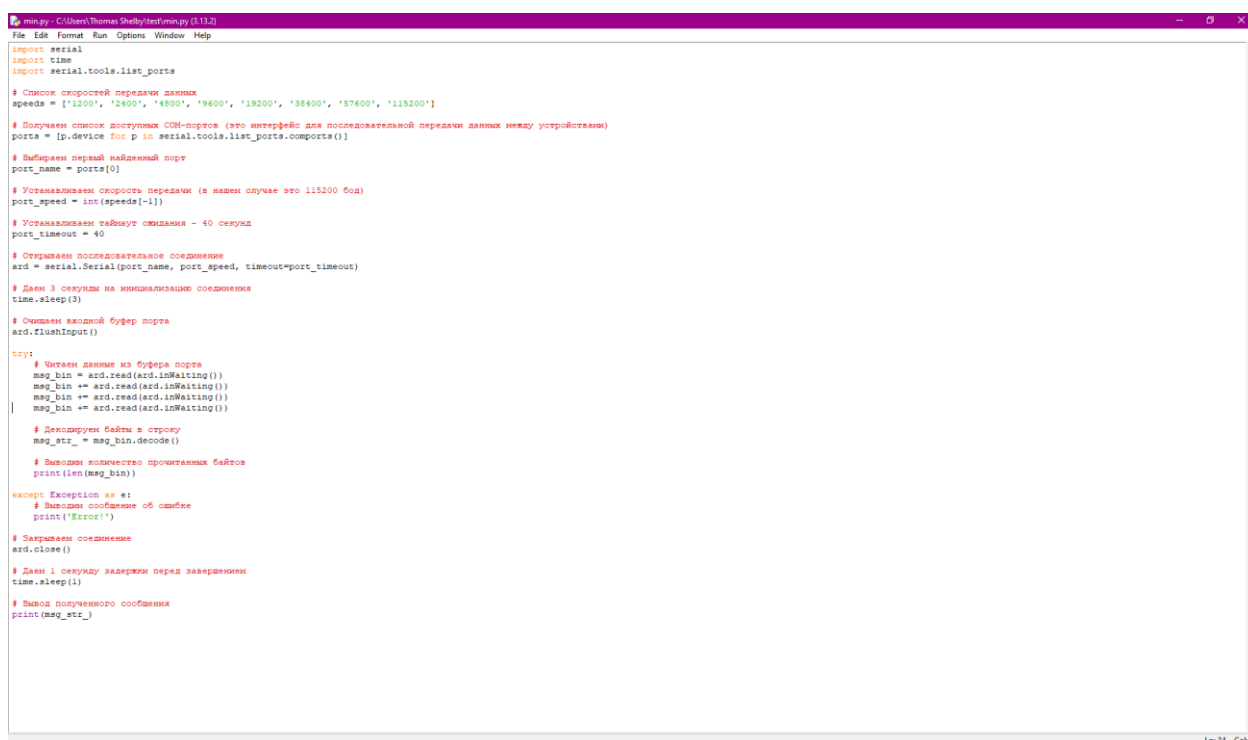
Реализация программы работы с последовательным портом средствами Python

Цель: закрепить навыки работы с Dockerfile, а также познакомиться с программой Anaconda.

Ход работы

Нам необходимо создать файл python, разместив там программу, которая показывает, какие свободные последовательные порты доступны.

Мы создаем файл с помощью IDLE python, в котором вписываем вышеописанную программу, попутно добавляя объяснение к каждой строке кода (рис. 1):



```
python - C:\Users\Thomas Shelby\testmypy (3.11.2)
File Edit Format Run Options Window Help
import serial
import time
import serial.tools.list_ports

# Список скоростей передачи данных
speeds = ['1200', '2400', '4800', '9600', '19200', '38400', '57600', '115200']

# Получаем список доступных COM-портов (это интерфейс для последовательной передачи данных между устройствами)
ports = [p.device for p in serial.tools.list_ports.comports()]

# Выбираем первый найденный порт
port_name = ports[0]

# Устанавливаем скорость передачи (в нашем случае это 115200 бод)
port_speed = int(speeds[-1])

# Устанавливаем таймаут ожидания - 40 секунд
port_timeout = 40

# Создаем последовательное соединение
ard = serial.Serial(port_name, port_speed, timeout=port_timeout)

# Даем 3 секунды на инициализацию соединения
time.sleep(3)

# Очищаем входной буфер порта
ard.flushInput()

try:
    # Читаем данные из буфера порта
    msg_bin = ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())

    # Декодируем байты в строку
    msg_str_ = msg_bin.decode()

    # Выводим количество прочитанных байтов
    print(len(msg_bin))

except Exception as e:
    # Выводим сообщение об ошибке
    print('Error!')

# Закрываем соединение
ard.close()

# Даем 1 секунду задержки перед завершением
time.sleep(1)

# Вывод полученного сообщения
print(msg_str_)
```

Рисунок 1 – программа

Далее нам нужно упаковать программу в Docker контейнер. Делаем следующее:

1. Устанавливаем Docker Desktop на свой компьютер, если не установлено

2. Пишем в PowerShell «notepad Dockerfile», нам открывается блокнот этого файла, в который мы вписываем (рис. 1.1).

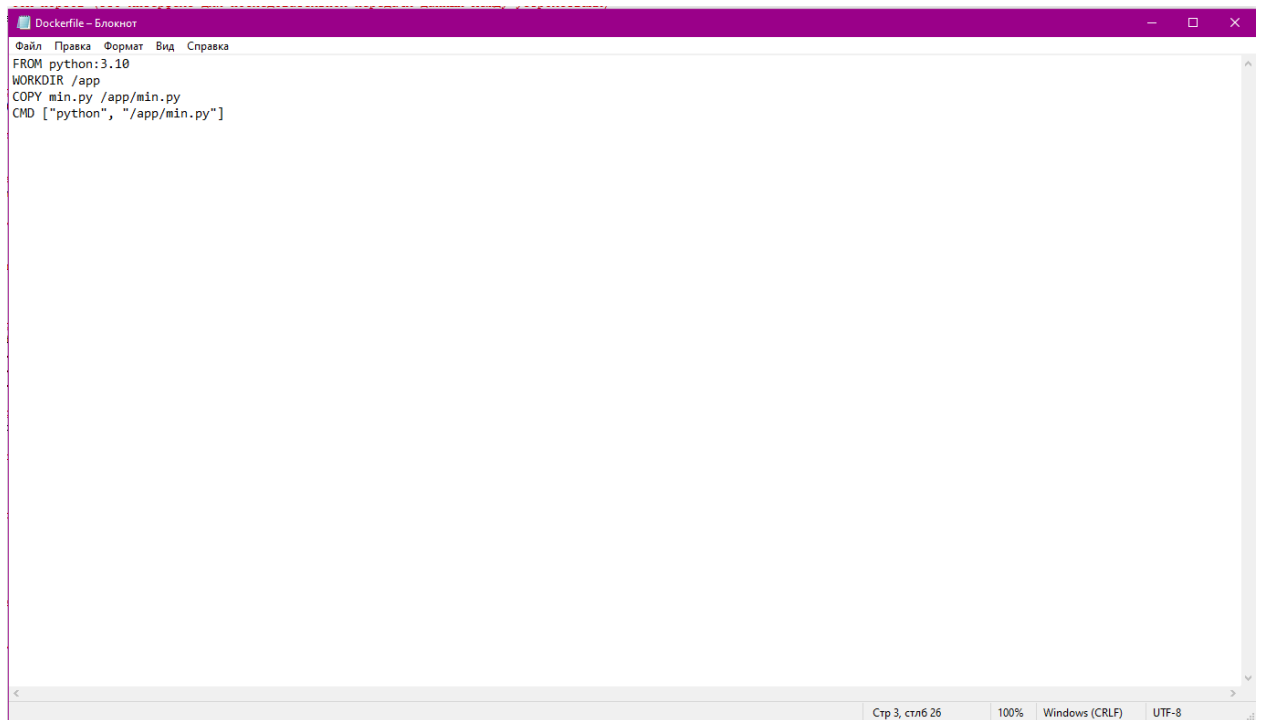


Рисунок 1.1 – Dockerfile

Чтобы не было проблем, закидываем файл min.py в одну папку с докером, потому что у меня лично, были ошибки.

3. Далее мы запускаем билд следующей командой:

docker build -t dockerfile2 .

4. Далее мы запускаем Docker-контейнер командой:

docker run -d --name dockerfile3 dockerfile2

5. После проверяем логи контейнера, используя команду:

docker logs

e32d023b8783198c80bf23b0b311db88b76aa5db8821b79136d7b14b0fb706a3

6. Как показано на рисунке 1.2, Docker грубо говоря зашел в файл, но не поймет, что с ним делать.

Вывод: в результате практической работы мы закрепили навыки работы с Dockerfile, изучили Anaconda и построили там с помощью jupyter Notebook график случайных чисел. Поэтапные скриншоты работ находятся в репозитории, также там логи PowerShell.