

## Практическая работа №3

**Тема:** «Реализация программы работы с последовательным портом средствами Python. Основы Anaconda и Jupyter Notebook»

**Ход работы:**

## 1. Установка библиотеки pyserial.

```
Microsoft Windows [Version 10.0.19045.5487]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\vika->pip install serial
Collecting serial
  Downloading serial-0.0.97-py2.py3-none-any.whl.metadata (889 bytes)
Collecting future>=0.17.1 (from serial)
  Downloading future-1.0.0-py3-none-any.whl.metadata (4.0 kB)
Collecting pyyaml>=3.13 (from serial)
  Downloading PyYAML-6.0.2-cp311-cp311-win_amd64.whl.metadata (2.1 kB)
Collecting iso8601>=0.1.12 (from serial)
  Downloading iso8601-2.1.0-py3-none-any.whl.metadata (3.7 kB)
Download serial-0.0.97-py2.py3-none-any.whl (40 kB)
----- 40.9/40.9 kB 140.1 kB/s eta 0:00:00
Download future-1.0.0-py3-none-any.whl (491 kB)
----- 491.3/491.3 kB 769.7 kB/s eta 0:00:00
Download iso8601-2.1.0-py3-none-any.whl (7.5 kB)
Download PyYAML-6.0.2-cp311-cp311-win_amd64.whl (161 kB)
----- 162.0/162.0 kB 607.0 kB/s eta 0:00:00
Installing collected packages: pyyaml, iso8601, future, serial
  WARNING: The scripts futurize.exe and pasteurize.exe are installed in 'C:\Users\vika-\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\Local
  h is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed future-1.0.0 iso8601-2.1.0 pyyaml-6.0.2 serial-0.0.97

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: C:\Users\vika-\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
```

2. Создание новой программы со следующим кодом и добавлением комментариев к каждой строке.

```
# Импортируем библиотеку для работы с последовательным портом
import serial

# Импортируем модуль time для работы с задержками
import time

# Импортируем инструменты для работы с портами компьютера
import serial.tools.list_ports

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

# Получаем список всех доступных последовательных портов
ports = [p.device for p in serial.tools.list_ports.comports()]

# Выбираем первый доступный порт из списка
port_name = ports[0]

# Выбираем максимальную скорость из списка скоростей
port_speed = int(speeds[-1])

# Устанавливаем таймаут для операций с портом
```

```

port_timeout = 10

# Создаем объект для работы с последовательным портом
ard = serial.Serial(port_name, port_speed, timeout = port_timeout)

# Делаем задержку в 1 секунду для стабилизации соединения
time.sleep(1)

# Очищаем входной буфер порта
ard.flushInput()

# Начинаем блок обработки исключений
try:

    # Читаем все доступные данные из порта
    msg_bin = ard.read(ard.inWaiting())

    # Читаем дополнительные данные, если они появились
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())

    # Преобразуем бинарные данные в строку
    msg_str_ = msg_bin.decode()

    # Выводим длину полученных бинарных данных
    print(len(msg_bin))

# Обрабатываем возможные ошибки
except Exception as e:
    print('Error!')

# Закрываем соединение с портом
ard.close()

# Делаем небольшую задержку перед выводом
time.sleep(1)

# Выводим полученную строку
print(msg_str_)

```

### 3. Проверка работы программы.

```

Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
import serial

```

```

===== RESTART: C:/Users/vika-/Desktop/3.py =====
0

```

### 4. Создание Docker контейнера.

```


=> => resolve docker.io/library/python:3.11@sha256:68a8863d0625f42d47e0684f33ca02f19d6094ef859a8af237aaf645195ed477 0.0s
ef859a8af237aaf645195ed477
0.0s
=> => sha256:68a8863d0625f42d47e0684f33ca02f19d6094ef859a8af237aaf645195ed477 9.08kB / 9.08kB
0.0s
=> => sha256:78f51bfc9d01014323cbc0d8d1f8ef4ed2b4ebfabef845b09b395950fb1f5428 2.33kB / 2.33kB
0.0s
=> => sha256:1d281e50d3e435595c266df06531a7e8c2ebb0c185622c8ab2eed8d760e6576b 64.39MB / 64.39MB
11.7s
=> => sha256:78553a4d82cb4da60dd65fbade310bee7c4c86e7575e4bbbb208b2ea089402ad 6.18kB / 6.18kB
[+] Building 52.2s (9/9) FINISHED docker:desktop-linux
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 300B 0.0s
=> [internal] load metadata for docker.io/library/python:3.11 2.6s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 2.47kB 0.0s
=> [1/3] FROM docker.io/library/python:3.11@sha256:68a8863d0625f42d47e0684f33ca02 45.0s
=> => resolve docker.io/library/python:3.11@sha256:68a8863d0625f42d47e0684f33ca02f 0.0s
=> => sha256:68a8863d0625f42d47e0684f33ca02f19d6094ef859a8af237aaf 9.08kB / 9.08kB 0.0s
=> => sha256:78f51bfc9d01014323cbc0d8d1f8ef4ed2b4ebfabef845b09b395 2.33kB / 2.33kB 0.0s
=> => sha256:1d281e50d3e435595c266df06531a7e8c2ebb0c185622c8ab2 64.39MB / 64.39MB 11.7s
=> => sha256:78553a4d82cb4da60dd65fbade310bee7c4c86e7575e4bbbb208b 6.18kB / 6.18kB 0.0s
=> => sha256:155ad54a8b2812a0ec559ff82c0c6f0f0dddb337a226b11879 48.48MB / 48.48MB 22.8s
=> => sha256:8031108f3cda87bb32f090262d0109c8a0db99168050967bece 24.06MB / 24.06MB 5.0s
=> => sha256:447713e77b4fc3658cfba0c1e816b70ff6d9bf06563dc8cf 211.34MB / 211.34MB 34.0s
=> => sha256:441749a24fb5824edbe40f9c29efb7f0eebf4607bba664b8a52a 6.16MB / 6.16MB 12.9s
=> => sha256:ae604eab20d6af7fea1faa7a0a2217413bb6f48d6865a062dc 24.31MB / 24.31MB 18.2s
=> => sha256:672d84e58157369c80f62fff5af0bf13c729ff624072006d6db30b11 251B / 251B 18.8s
=> => extracting sha256:155ad54a8b2812a0ec559ff82c0c6f0f0dddb337a226b11879f09e15f6 3.4s
=> => extracting sha256:8031108f3cda87bb32f090262d0109c8a0db99168050967becefad502e 0.9s
=> => extracting sha256:1d281e50d3e435595c266df06531a7e8c2ebb0c185622c8ab2eed8d760 4.0s
=> => extracting sha256:447713e77b4fc3658cfba0c1e816b70ff6d9bf06563dc8cfcb0459406a 8.8s
=> => extracting sha256:441749a24fb5824edbe40f9c29efb7f0eebf4607bba664b8a52a2597c2 0.4s
=> => extracting sha256:ae604eab20d6af7fea1faa7a0a2217413bb6f48d6865a062dc6e268cab 1.2s
=> => extracting sha256:672d84e58157369c80f62fff5af0bf13c729ff624072006d6db30b116d 0.0s
=> [2/3] COPY 3.py . 0.5s
=> [3/3] RUN pip install pyserial==3.5 3.8s
=> => exporting to image 0.2s
=> => exporting layers 0.1s
=> => writing image sha256:36b50aab336581c462be04c7ebee6a4710c4925d205c7ed4f1d09f 0.0s
=> => naming to docker.io/library/my-app 0.0s

vika-@DESKTOP-FP6MJE7 MINGW64 ~/Desktop/lab3
$ docker save -o 3.tar 3

```


## 5. Размещение проекта в собственном репозитории.


ist /



Drag additional files here to add them to your repository

Or [choose your files](#)

 3.py

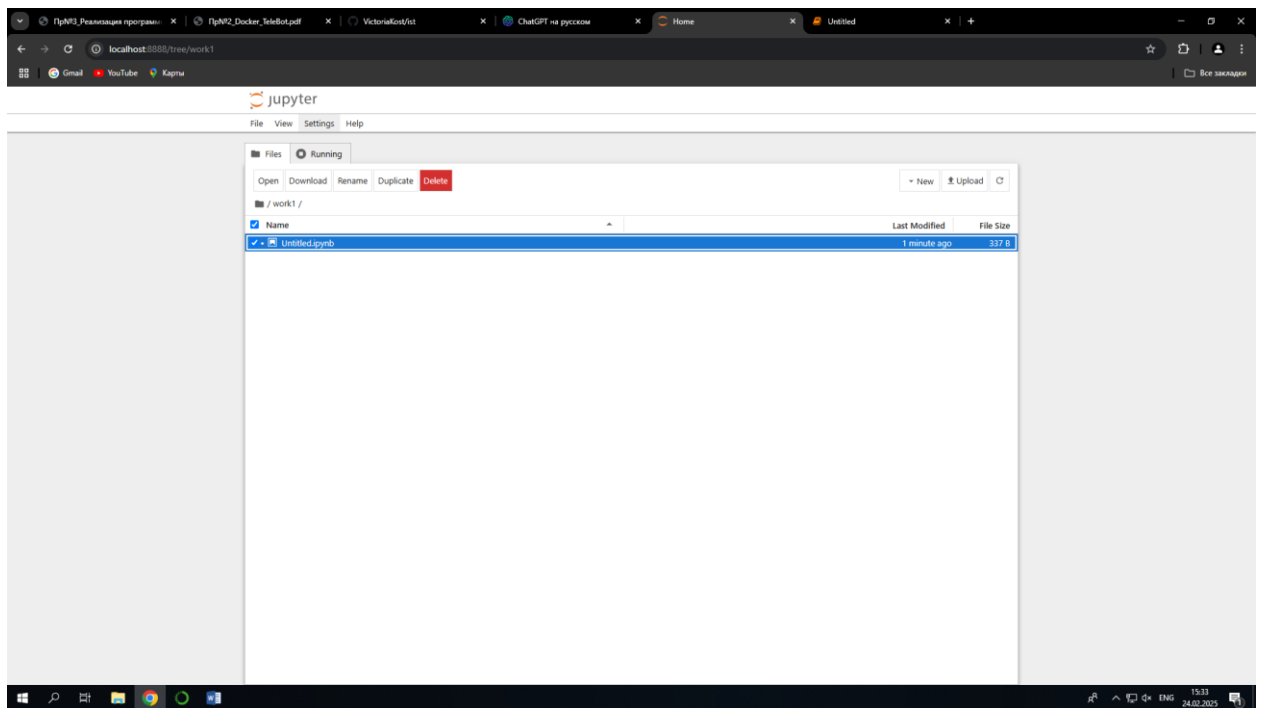


**Commit changes**

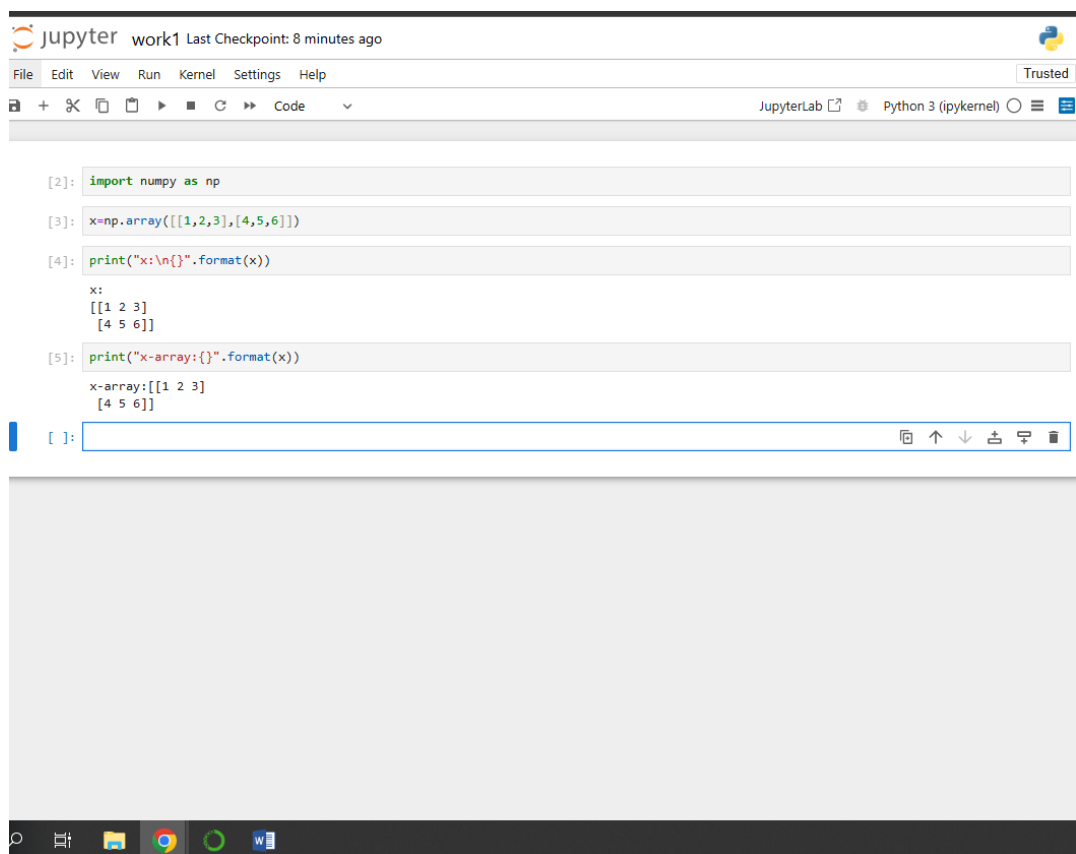
третья практика

Add an optional extended description...

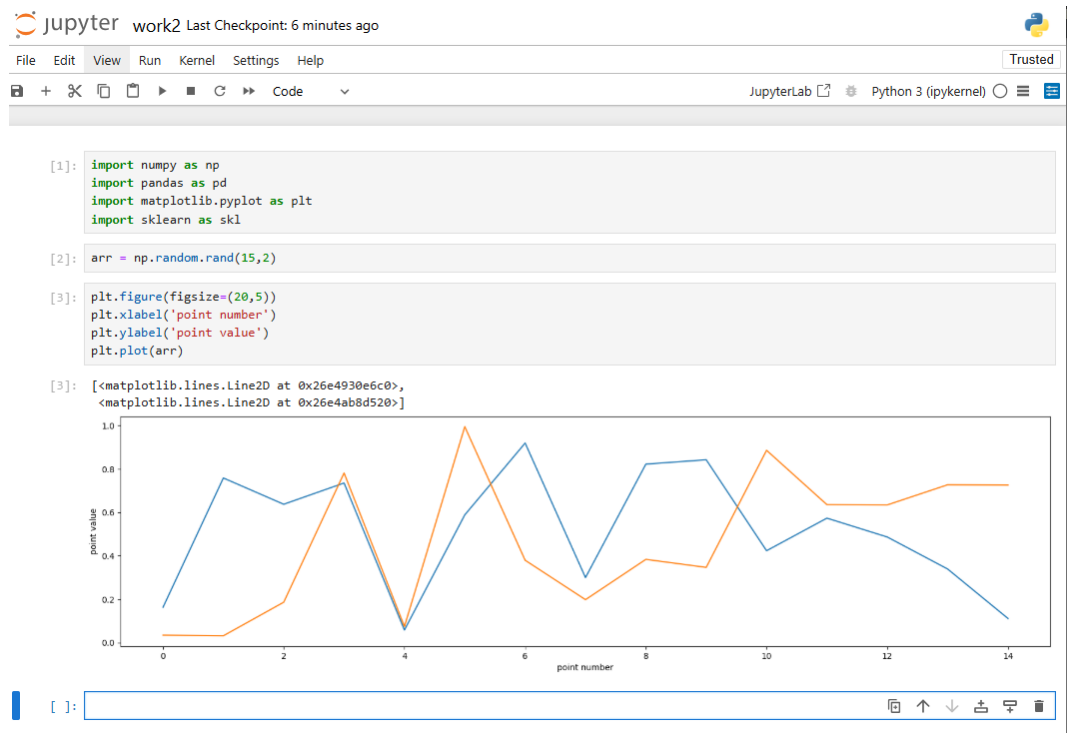
## 6. Создание новых папки и файла в Jupyter Notebook



## 7. Написание кода программы 1.



## 8. Проверка результатов работы программы 2.



**Вывод:** в ходе выполнения практической работы были успешно освоены основные принципы работы с последовательным портом при помощи Python и библиотеки pyserial, а также освоены базовые функции Jupyter Notebook. Все поставленные задачи выполнены в полном объеме, программа работает корректно.