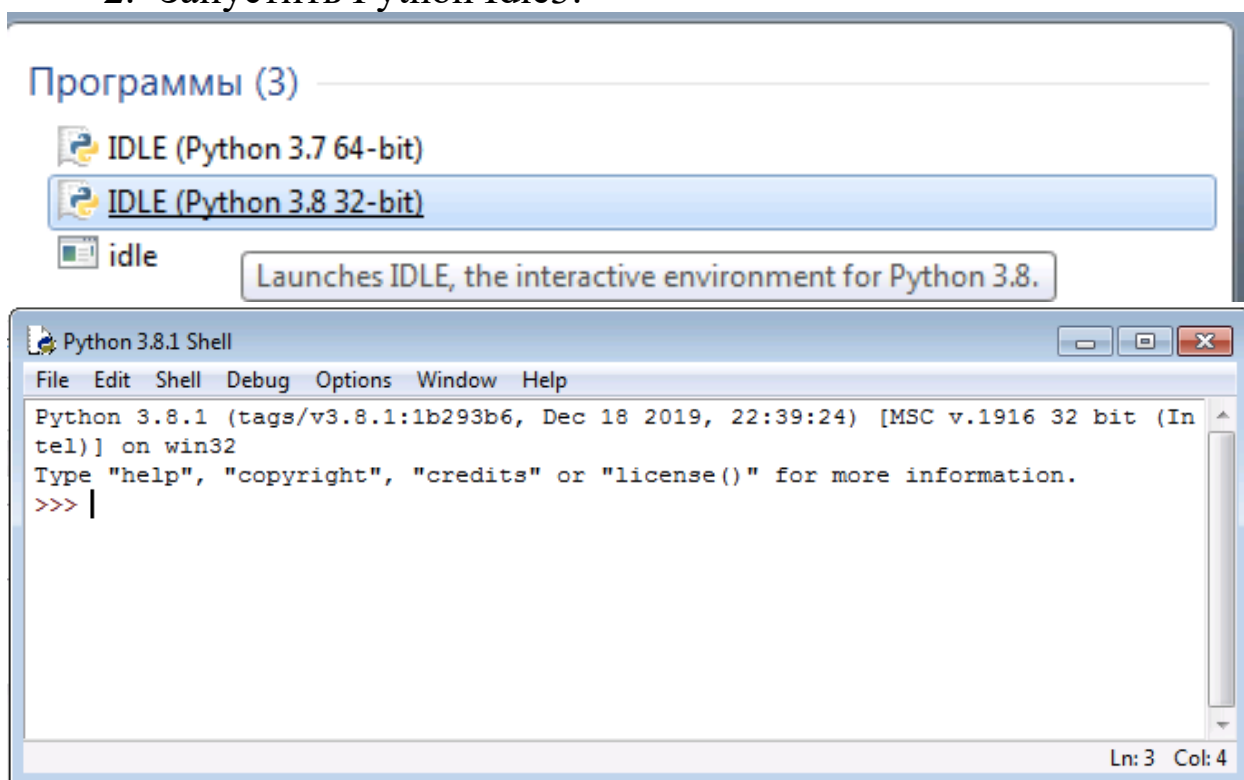


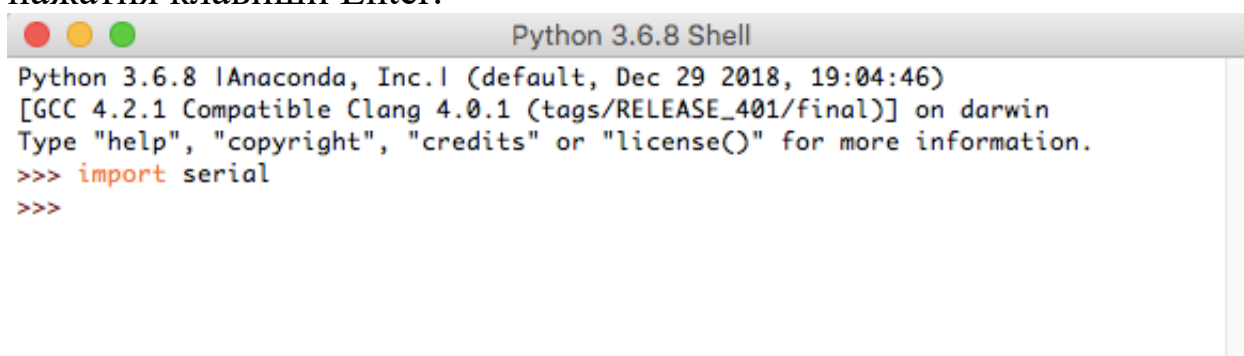
Реализация программы работы с последовательным портом средствами Python

Пояснения

0. Подготовка к работе: изучить https://en.wikipedia.org/wiki/Serial_port
1. Установить среду программирования Python (<https://www.python.org/downloads/>).
2. Запустить Python Idle3.



3. Проверить наличие необходимой библиотеки *pyserial*. Обратите внимание, что в данной среде команды выполняются по одной (Python – интерпретируемый язык программирования), после нажатия клавиши Enter.



4. Возникновение ошибки свидетельствует об отсутствии соответствующей библиотеки в текущей версии Python. Для установки:

4.1. Закрыть Python Idle

4.2. Запустить командную строку ОС.

4.4. Установить библиотеку pyserial для чего набрать команду
pip3 install pyserial

или

pip install serial

или

python -m pip install pyserial

5. Повторить с п. 2. В случае повторения ошибки изучить <https://pyserial.readthedocs.io/en/latest/pyserial.html> и реализовать.

6. Создать новую многострочную программу Python (Python Idle – File – New File).

7. Опробовать программу со следующим текстом.

```
import serial
import time
import serial.tools.list_ports
speeds = ['1200', '2400', '4800', '9600', '19200', '38400', '57600', '115200']
ports = [p.device for p in serial.tools.list_ports.comports()]
port_name = ports[0]
port_speed = int(speeds[-1])
port_timeout = 10
ard = serial.Serial(port_name, port_speed, timeout = port_timeout)
time.sleep(1)
ard.flushInput()
try:
    msg_bin = ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_str_ = msg_bin.decode()
    print(len(msg_bin))
except Exception as e:
    print('Error!')
ard.close()
time.sleep(1)
print(msg_str_)
```

8. Подготовить комментарии к каждой строке программы.
9. Упаковать программу в Docker контейнер по аналогии с ч. 2.
10. Разместить проект в собственном репозитории Github по аналогии с ч. 1.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое последовательный порт и для чего он в настоящий момент используется?
2. Какие версии Python поддерживают работу с pyserial?

Основы Anaconda и Jupyter Notebook

Одним из наиболее распространенных средств создания AI-систем является язык программирования Python. Стоит учитывать наличие значительного количества сред программирования для данного ЯП. Наиболее распространенные:

- Python Idle – стандартная среда программирования, не требует значительных ресурсов, используется для отладки программ «на месте» (например, в различных micro-PC таких как Raspberry Pi, Orange Pi и т.д.);

- Spyder – более функциональная среда по сравнению с Python Idle, используется для разработки и первично отладки программ на языке Python;

- Jupyter Notebook – среда разработки, апробации и первичной отладки алгоритмов интеллектуальной обработки данных (например, апробации различных вариантов искусственных НС для последующего встраивания их в общую программу).

Для использования всех указанных выше инструментов необходимо установить дистрибутив языков программирования Python и R, включающий набор популярных свободных библиотек, объединённых проблематикой науки о данных и машинного обучения (<https://www.anaconda.com/download/#windows> или google -> anaconda download).

Процесс установки достаточно прост и не отличается от установки любой другой программы. После установки в списке программ появится Anaconda Navigator (рис.).

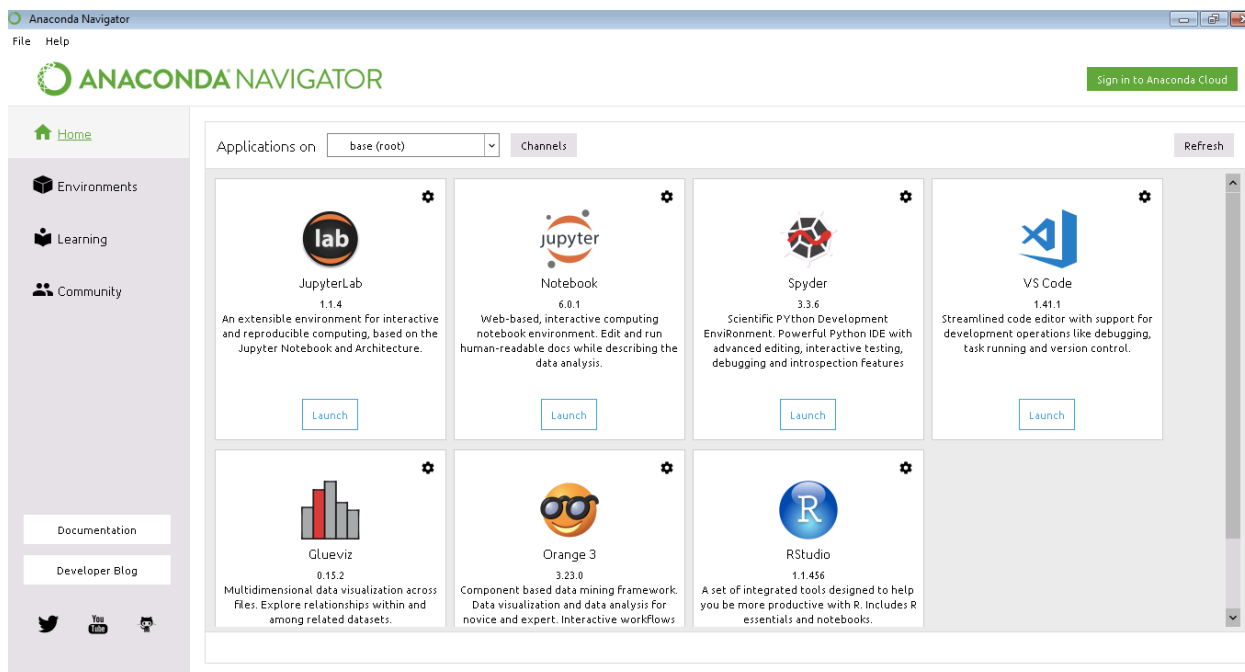


Рисунок – Anaconda Navigator

Запускаем jupyter Notebook (“Launch”) – интерфейсом программирования будет Ваш браузер (рис.).

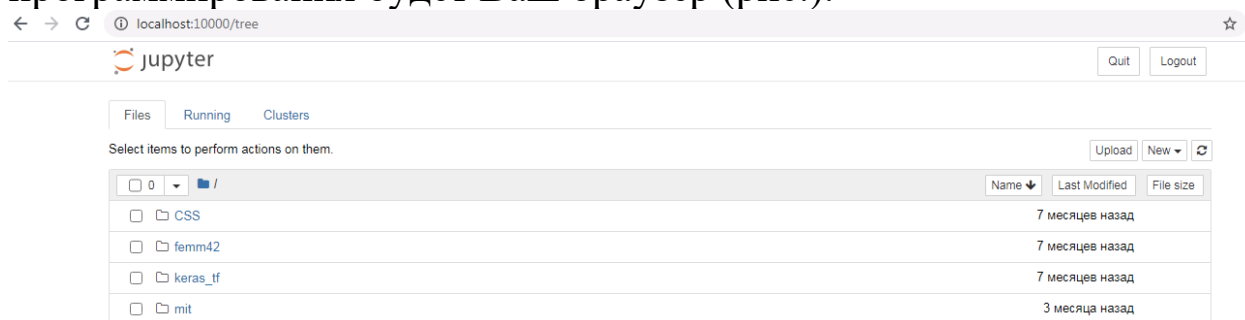
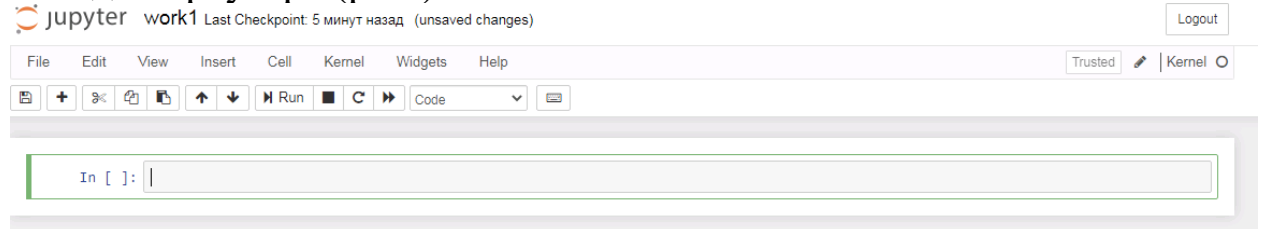


Рисунок – Jupyter Notebook (директории зависят от Вашего компьютера)

1. В правом верхнем углу нажать – “New” → “Folder”. В списке появится “Untitled Folder” (папка без названия).
2. Поставить галочку слева от имени “Untitled Folder”.
3. В левом верхнем углу появится кнопка “Rename” (переименовать) → вводим имя каталога (work1).
4. Нажимаем на новый каталог.
5. “New” → “Python 3”. Открывается новая вкладка браузера с файлом “Untitled” (.ipynb). Закрываем вкладку.
6. В каталоге work1 видим файл Untitled.ipynb ставим возле него галочку → нажимаем на появившуюся вверху слева кнопку “Shutdown” (остановить выполнение).

7. Снова файл Untitled.ipynb ставим возле него галочку → нажимаем на появившуюся сверху слева кнопку “Rename” → вводим имя файла (work1.ipynb).

8. Нажимаем на файл work1.ipynb открывается окно в новой вкладке браузера (рис.).



Рисунок– Jupyter Notebook work1.ipynb

8. Текст программ в Jupyter Notebook разделен на блоки. Каждый блок выполняется отдельно – для этого нажимается сочетание клавиш Shift + Enter (рис.). После выполнения блока ему присваивается номер (“In [1]”) и автоматически создается новый блок (также его можно создать кнопкой “+” в верхней панели инструментов). Результат сохраняется в оперативной памяти.

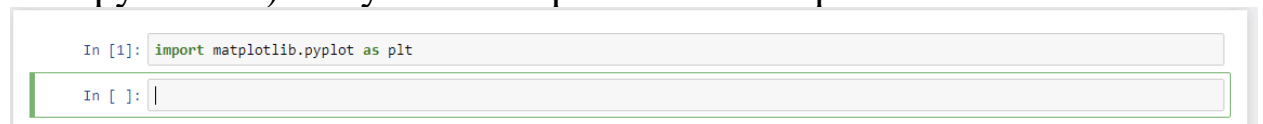


Рисунок – Jupyter Notebook work1.ipynb

9. Можно вернуться к выполненному блоку и выполнить его повторно (номер блока инкрементируется, рис.).

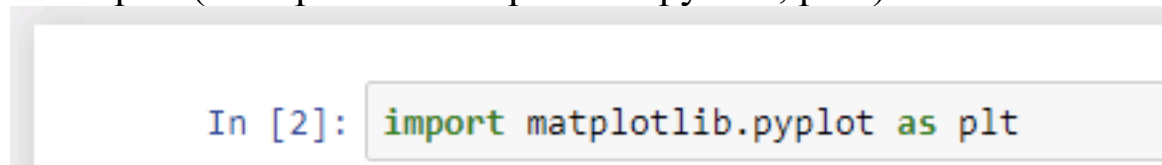



Рисунок – Jupyter Notebook work1.ipynb

10. Очистка оперативной памяти для данной программы производится перезагрузкой ядра (кнопка  на панели инструментов).

11. Разделение на блоки удобно, когда требуется провести анализ обработки одних и тех же данных различными алгоритмами (рис.).

12. Необходимо проверить наличие необходимых библиотек (рис.).

```

In [1]: import numpy as np # импорт библиотеки нужен, чтобы можно было работать с ее функциями
        # импортируется каждая библиотека отдельно, чтобы все не занимало слишком много ОЗУ
        # библиотека numpy используется для быстрой работы с числами и массивами чисел
        # as np - далее в тексте программы будем ссылаться на эту библиотеку через имя np
        # np - общепринятый акроним numpy

In [2]: x = np.array([[1, 2, 3], [4, 5, 6]]) # создаем numpy-массив 2x3

In [3]: print("x:\n{}".format(x)) # выводим значения массива с отделением его имени от значений знаком \n - перенос на новую строку

x:
[[1 2 3]
 [4 5 6]]

In [4]: print("x-array:{}".format(x)) # выводим значения массива без отделением его имени от значений

x-array:[[1 2 3]
 [4 5 6]]

In [ ]:

```

Рисунок – Jupyter Notebook work1.ipynb

```

In [1]: import numpy as np # work with numbers
        import pandas as pd # work with various data
        import matplotlib.pyplot as plt # work with graphs
        import sklearn as skl # machine learning library

```

Рисунок – Проверка наличия библиотек (должно выполняться без ошибок)

```

In [2]: arr = np.random.rand(15,2) # создаем массив случайных чисел - 15x2

In [3]: plt.figure(figsize=(20,5)) # создаем область для рисования размером 20x5
        plt.xlabel('point number') # задаем название горизонтальной оси
        plt.ylabel('point value') # задаем название вертикальной оси
        plt.plot(arr) # выводим графики на экран

Out[3]: [<matplotlib.lines.Line2D at 0xa146352e8>,
         <matplotlib.lines.Line2D at 0xa146354e0>]

```

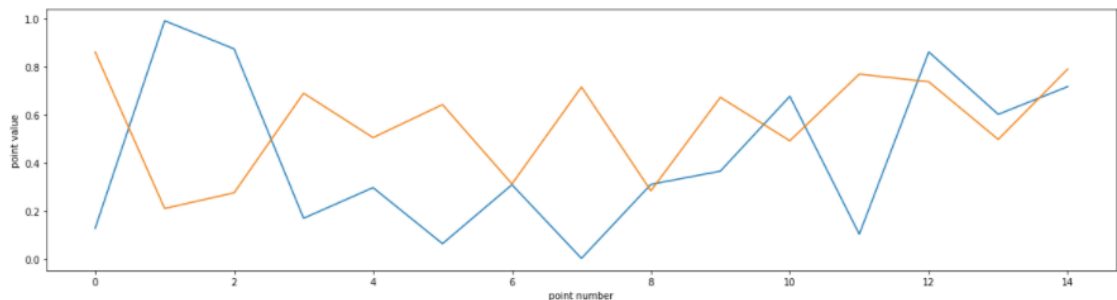


Рисунок – Программа для выполнения и отчета