Практикум 1. Пошаговые вычисления в пакете ANACONDA.

Цель работы – знакомство с интерфейсом, представлением различных типов числовых данных и действиями над ними.

Продолжительность работы - 2 часа.

Оборудование, приборы, инструментарий – работа выполняется в компьютерном классе с использованием пакета Anaconda.

Порядок выполнения

1. Упражнения выполняются параллельно с изучением теоретического материала.
1. После выполнения каждого упражнения результаты заносятся в отчёт.
2. При выполнении упражнений в случае появления сообщения об ошибке рекомендуется сначала самостоятельно выяснить, чем оно вызвано, и исправить команду; если многократные попытки устранить ошибку не привели к успеху, то проконсультироваться с преподавателем.
3. Дома доделать упражнения из раздела «Краткие теоретические сведения и практические упражнения», которые Вы не успели выполнить во время аудиторного занятия.
4. После выполнения упражнений выполнить дополнительные упражнения для самостоятельной работы и ответить на контрольные вопросы и (см. ниже).
5. Подготовить **отчёт,** в который включить упражнения из раздела «Краткие теоретические сведения и практические упражнения» и упражнения для самостоятельной работы. Отчёт представить либо в формате интерактивного питон-скрипта (ipynb), либо в виде документа Microsoft Word. Файл следует назвать по следующей схеме **pin_10_Ivanov_P_01_s1** (группа, фамилия, инициалы, номер лабораторной, семестр). Отчет должен содержать по каждому выполненному упражнению: № упражнения, текст упражнения; команды, скопированные из командного окна, с комментариями к ним и результаты их выполнения, включая построенные графики; тексты python фукнций; выводы.

Краткие теоретические сведения

и практические упражнения

1. Основные окна рабочего стола Anaconda.

После запуска пакета Anaconda загрузится навигатор и предложит на выбор одно из средств разработки (представлено на рисунке 1 ниже).

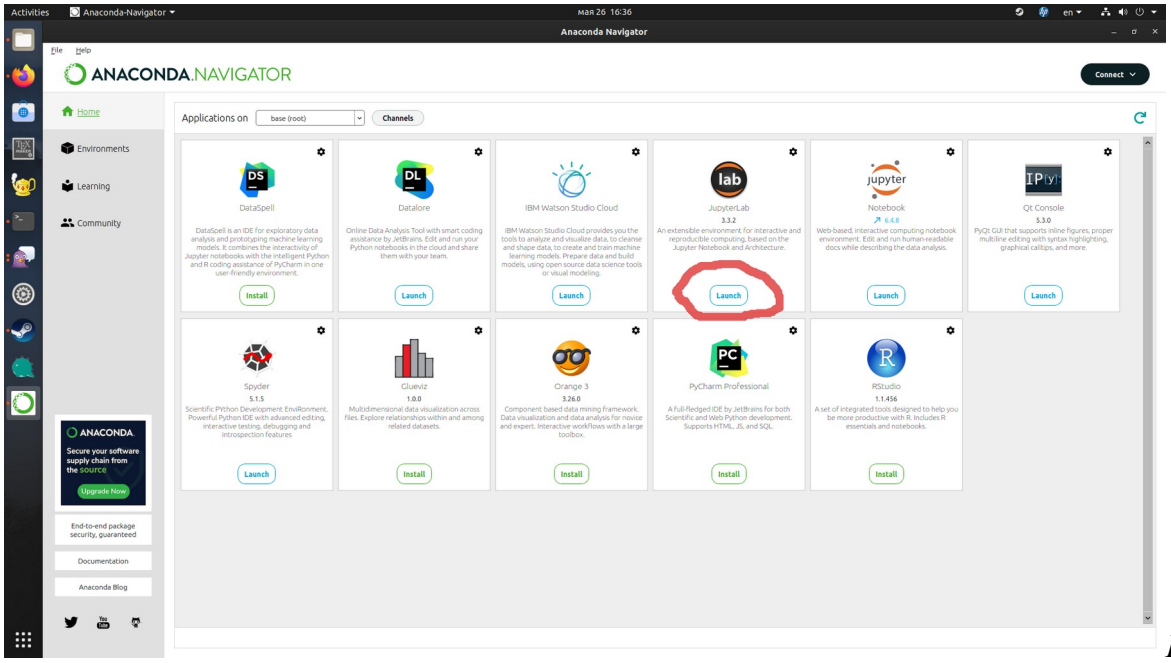

Рисунок 1: Навигатор Anaconda

Для выполнения лабораторных работ по умолчанию будем использовать **JupyterLab** (на рисунке 1 это четвертый блок в верхнем ряду). После загрузки рабочего пространства **JupyterLab** в отдельной вкладке браузера отобразится:

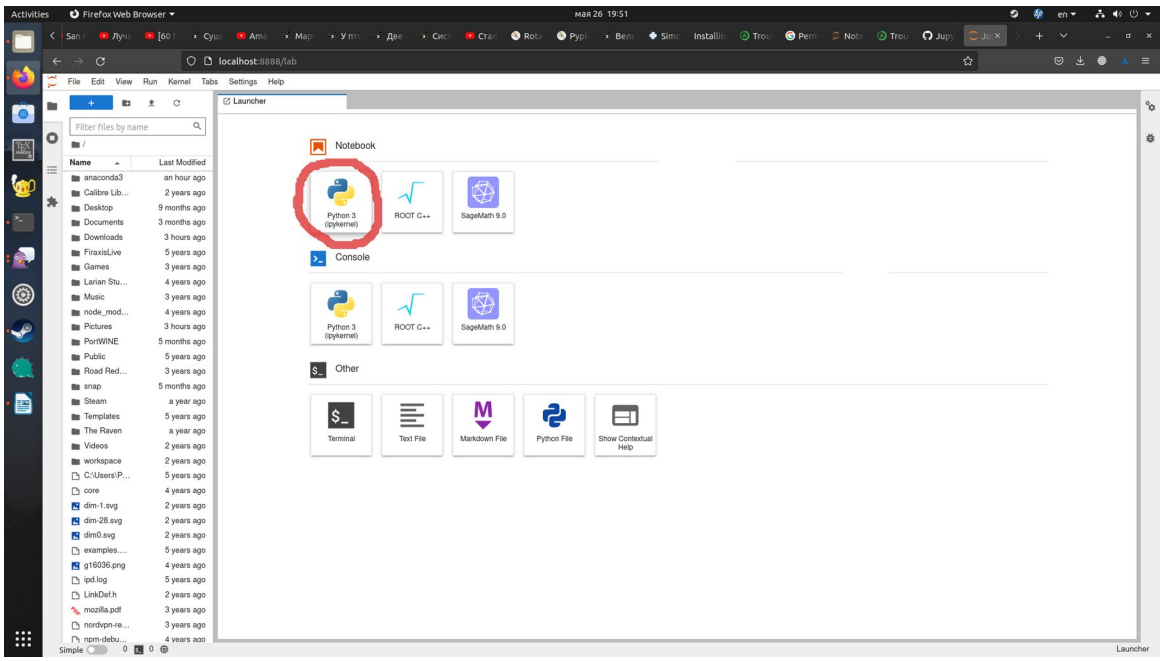

Рисунок 2: стартовое меню Jupyterlab

Среди этих опций выбираем python 3 (ipkernel) в разделе **Notebook** (обведено на рисунке 2). В результате будет создана новая интерактивная тетрадь Untitled1.

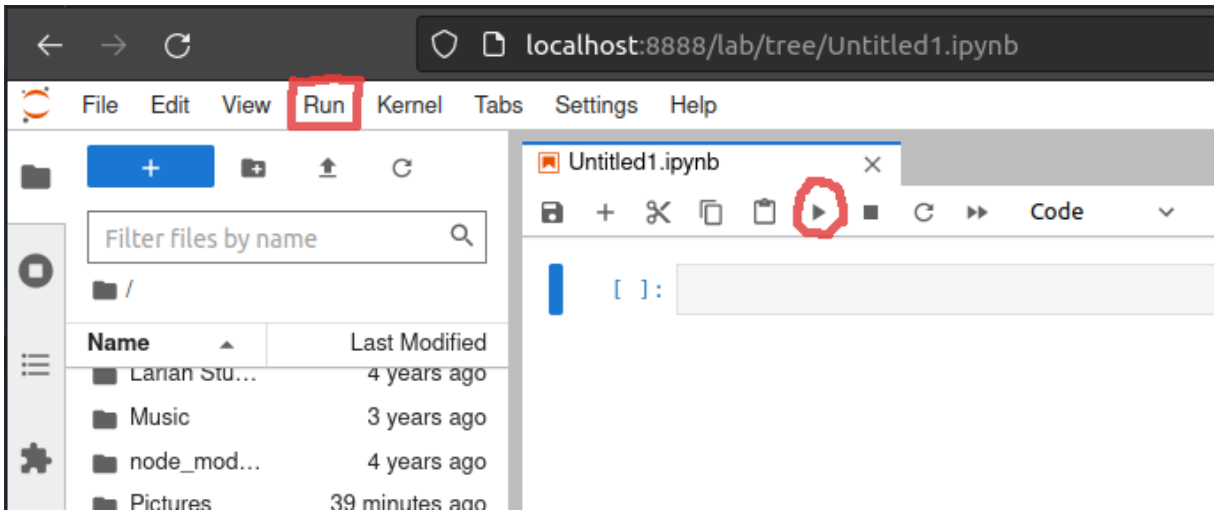

Рисунок 3: Рабочая тетрадь JupyterLab

Упражнение 1. Записать следующие команды (каждую в своей ячейке).

```
1)  2*3
2)  k=3+4
print(k)
3) (k+1)*(k-1)
4) (x+1)*(x-1)
```

Выполнить их по очереди, запуская ячейки либо через меню Run, либо с помощью пиктограммы треугольник (см. рис. 3).

5) Если формула для вычисления очень длинная, то ее можно перенести на следующую строку. Признаком завершения сроки, у которой имеется продолжение на следующей строке, являются обратный слеш (\ ).

```
h=(k+2)*3+\
3+(k+7)
print(h)
```

2. Переменные рабочего пространства.

В именах переменных можно использовать латинские буквы, цифры и символ подчеркивания; большие и малые буквы в именах различаются; имя должно начинаться с буквы; длина имени не должна превышать 63 символа.

Информацию о переменных рабочего пространства можно получить, набрав в отдельной ячейке команду **%whos**

Если в дальнейших вычислениях переменная h, к примеру, не понадобится, ее можно убрать из рабочего пространства, набрав в ячейке **del (a).**

Команда **%reset** удаляет все переменные (проверьте).

Упражнение 3.

1) Убрать из рабочего пространства все переменные.

2) Ввести переменные x, y, z, t, задав им значения соответственно 1, 2, 3, 4.

3) Вывести в командное окно информацию обо всех переменных.

4) Удалить из рабочего пространства переменную x.

5) Вывести в командное окно информацию об оставшихся переменных.

6) Вывести в командное окно информацию об оставшихся переменных.

3. Представление данных матрицами

Матрицей размерности $n \times m$ называется прямоугольная таблица, состоящая из *n* строк и *m* столбцов. Традиционно в математике эту таблицу заключают в круглые скобки. Например, $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 3 & 2 \end{pmatrix}$ - матрица размерности 2×4;

Если матрица имеет размер $1 \times m$, т.е. состоит только из одной строки, то ее называют вектором-строкой. Например, $B = (1 \quad 3 \quad -1)$ - матрица размерности 1×3, т.е. вектор-строка.

Если матрица имеет размер $n \times 1$, т.е. состоит только из одного столбца, то ее называют вектором-столбцом. Например, $C = \begin{pmatrix} -1 \\ 2{,}1 \end{pmatrix}$ - матрица размерности 2×1, т.е. вектор-столбец.

Если матрица имеет размер 1×1, т.е. состоит из одного столбца и одной строки, то ее называют скаляром. Например, $D = (9)$ - матрица размерности 1×1, т.е. скаляр.

В первую очередь рассмотрим как в рабочей тетради **Jupyterlab** можно отображать матрицы в составе текстовых ячеек. Для этого создадим новую ячейку и переключим её из режима **Code** в режим **Markdown** (см. верхнюю панель рисунка 3). Затем введите в этой ячейке следующее выражение:

```
$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 0 & 1 \\ 0 & 2 & 4 \end{bmatrix}$
```

Запустив на исполнение содержимое ячейки мы получим отформатированное изображение матрицы. Данный язык разметки для ввода формул (они

должны быть заключены между двумя знаками: $ $) является реализацией **Latex** и будет нами использоваться для описания заданий и развернутых комментарий с математическими формулами к ним. В той же ячейке, где мы с вами вывели изображение этой матрицы мы могли добавить свой текст перед или после матрицы.

Если же мы желаем не просто красиво нарисовать матрицу, а собственно ввести её в python и провести с ней некоторые вычисления, то нам потребуется в следующей ячейке снова перейти в режим **Code** из режима **Markdown**. Введите в этой ячейке следующий код:

```
import numpy as np
B = np.array([1, 3, -1])
print(B)
```

В этом примере мы подключили модуль **numpy** и назвали его для краткости «**np**» в нашей программе. Для задания вектора-строки мы использовали встроенный тип **array** из модуля **numpy**. Точно также можно задать матрицу, число строк и столбцов которой больше одного (двумерный массив):

```
import numpy as np
A = np.array( [ [1, 2, 3, 4], [0, 1, 3, 2] ] )
print(A)
```

Для доступа к отдельным элементам матриц указываются их индексы, причем нумерация индексов начинается с нуля. Например, A[1,3] – элемент матрицы A, стоящий в 2-й строке и 4-м столбце; B[2] – третий элемент вектора-строки B. Для доступа целиком к первой строке матрицы A нужно использовать нотацию вида: `A[:,1]`

Далее мы будем часто использовать векторы-строки, элементы которых образуют арифметическую прогрессию. Они задаются следующим образом:

```
v = np.arange(1,8,2)
print (v)
```

В этом примере мы указываем начальное значение «1», предельное «8» (это значение не включается в массив) и шаг «2» с которым нужно построить массив.

Если функции arange указать только один параметр, то она посчитает это значение предельным, начальный элемент нулевым, а шаг единичным.

```
import numpy as np
A = np.arange(4)
```

```
print('A =', A)
B = np.arange(12).reshape(2, 6)
print('B =', B)
```

В этом примере мы продемонстрировали работу функции **reshape**, которая позволяет сделать из вектора длины 12 двумерный массив нужной нам размерности (2 на 6). Все операции с размерностями представлены в таблице 1.

Таблица 1.	
Функция	Выполняемое действие
`np.shape(A)`	Возвращает массив размерностей матрицы A (для матрицы 2x3 вернет массив [2,3]).
`A.reshape(n,k)`	Превратит массив A в матрицу размерности n на k
`A.size`	Возвращает общее число элементов в матрице A
`len(A)`	Возвращает длину массива A

Элементами матриц могут быть любые выражения, допустимые в **python**.

```
import numpy as np
S = np.array([-1, np.sqrt(2), np.abs(-3)])
print (S)
```

Упражнение 4.

1) Задать какую-нибудь матрицу R размерностью 3×4. Отобразить её сначала в текстовой ячейке в режиме **Markdown**. Затем ввести её в кодовой ячейке с помощью модуля **numpy**.

2) Заменить значения элемента R(2,3) на противоположный (R(2,3)= -R(2,3)), вывести обновленную матрицу R в командное окно. Уменьшить на 4 элемент, стоящий в первой строке и третьем столбце, вывести обновленную матрицу R в командное окно. Удвоить все элементы второго столбца. Утроить все элементы первой строки.

3) Задать векторы-строки размерности 1×5 и 1×7, задать 3 вектора-столбца разной размерности.

4. Основные математические функции

Все операции над массивами реализуются посредством функций. *С каждой из традиционных операций (с умножением, делением и возведением в степень) связаны по две функции.* Список этих функций приведен в табл. 2. Серым цветом

выделены функции, которыми будем пользоваться после изучения соответствующих понятий в курсе линейной алгебры.

Таблица 2. **Арифметические функции**	
Символ	Выполняемое действие
+	Покомпонентное сложение числовых массивов одинаковой размерности.
-	Покомпонентное вычитание числовых массивов одинаковой размерности.
*	Покомпонентное умножение массивов одинаковой размерности
`.matmul`	Умножение матриц в соответствии с правилами линейной алгебры (условие выполнения: число столбцов первого сомножителя должно быть равно числу строк второго сомножителя) **C = np.matmul(A,B)** Альтернатива: **np.dot()**
`np.linalg.inv`	Взятие обратной матрицы **B = np.linalg.inv(A)**.
/	Покомпонентное деление элементов массивов одинаковой размерности.
`.transpose`	Транспонирование матрицы B = A.transpose()

Также в подмодуле **matlib** содержатся функции для создания матриц из всех нулей **np.matlib.zeros((2,2))**, из всех единиц **np.matlib.ones((2,2))** и особых единичных матриц **np.matlib.eye((2,2))** (единицы там стоят на главной диагонали). Полученные матрицы M можно конвертировать в массивы numpy и обратно с помощью функций **a = np.asarray(M)** и **M = np.asmatrix(a)**. Пример создания матриц с помощью `matlib`:

```
import numpy as np
import numpy.matlib
X = np.matlib.ones((2,3))
Y = np.matlib.matrix('1, 2, 3; 3, 2, 1')
print(X)
print(Y)
```

Упражнение 5.

1) Ввести матрицы

$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$, $B = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 3 & 4 \end{pmatrix}$, $c = 2$, $D = ones(2,3)$, $E = eye(3,3)$.

2) Выполнить операции (или убедиться, что их выполнить нельзя): $A + B$, $A + c$, $A + E$ $A - B$, $A - c$, $c * A$, а также произведение матриц A и B (а также A на транспонированную матрицу B).

В таблице 3 приведен список основных элементарных функций.

Таблица 3. **Элементарные математические функции**	
Категория функций	Наименование функций
Тригонометрические, аргумент в радианах	np.sin(x), np.cos(x), np.tan(x)
Обратные тригонометрические, результат в радианах	np.arcsin(x), np.arccos(x), np.arctan(x)
Гиперболические функции	np.sinh(x), np.cosh(x)
Степени, логарифмы, корни	np.exp(x), np.log(x), np.sqrt(x)
Модуль числа	np.fabs(x)
Знак числа	np.sign(x)
Округление по обычным математическим правилам	np.around(x)

Подробную информацию о каждой функции можно получить с помощью команды **help (имя функции).** Например,

```
help (np.cos)
```

Очень важная особенность функций в python - обработка аргументов, заданных матрицами. Например,

```
import numpy as np
x = np.array([[1,3,4],[6,1,0]])
A=np.sin(x)
print(A)
```

Упражнение 7.

1) Вычислить $\sqrt{1}, \sqrt{3}, \sqrt{5}$ с помощью задания данных в виде вектора.

Задания для самостоятельной работы

1. Выполнить упражнения из раздела «Краткие теоретические сведения и практические упражнения», которые не успели сделать в аудитории.

2. Самостоятельно выполнить упражнения:

Упражнение С1.

1) Задать вектор-строку с элементами от -2 до 10 с шагом 2, утроить все ее элементы.

2) Задать вектор-строку с элементами от 45 до 5 с шагом -5, определить ее размерность.

Упражнение С2.

1) Вычислить значения $\cos(x)$ одновременно при $0, \frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}, ..., 2\pi$. То же для остальных тригонометрических функций (использовать значение np.pi).

2) Вычислить значение выражения $y = \mathrm{ch}^2(x) - \mathrm{sh}^2(x)$ одновременно при $x = -2, -1.5, -1, ..., 2$.

3. Повторить теоретический материал, ответить на контрольные вопросы:

1) Какие имена переменных являются допустимыми?
2) Каким образом можно получить информацию о переменных рабочего пространства?
3) Каким образом можно получить подробную информацию о функции или команде?
4) Как задать матрицу произвольной размерности?
5) Каким образом осуществляются поэлементные арифметические действия с матрицами одинаковой размерности?

Список рекомендуемой литературы

1. А. Кривелёв. Основы компьютерной математики с использованием системы MatLab. М, 2005. – 2.1, 3.1, 4.1.

2. В.Г.Потемкин "Введение в Matlab" (v 5.3), http://matlab.exponenta.ru/ml/book1/index.php - 1.1, 1.7, 8.6