# Лабораторный практикум №1. Знакомство с пакетом Anaconda

### Введение

Этот практикум представляет собой выполнение на компьютере лабораторных работ по темам, изучаемым в параллельном теоретическом курсе линейной алгебры и аналитической геометрии. Главная цель практикума - научится решать математические задачи с помощью программных пакетов, автоматизирующих математические вычисления.

Для математических вычислений мы будем использовать язык Python (название произносится как Пайтон или Питон) с дополнительными библиотеками NumPy, SymPy, Matplotlib.

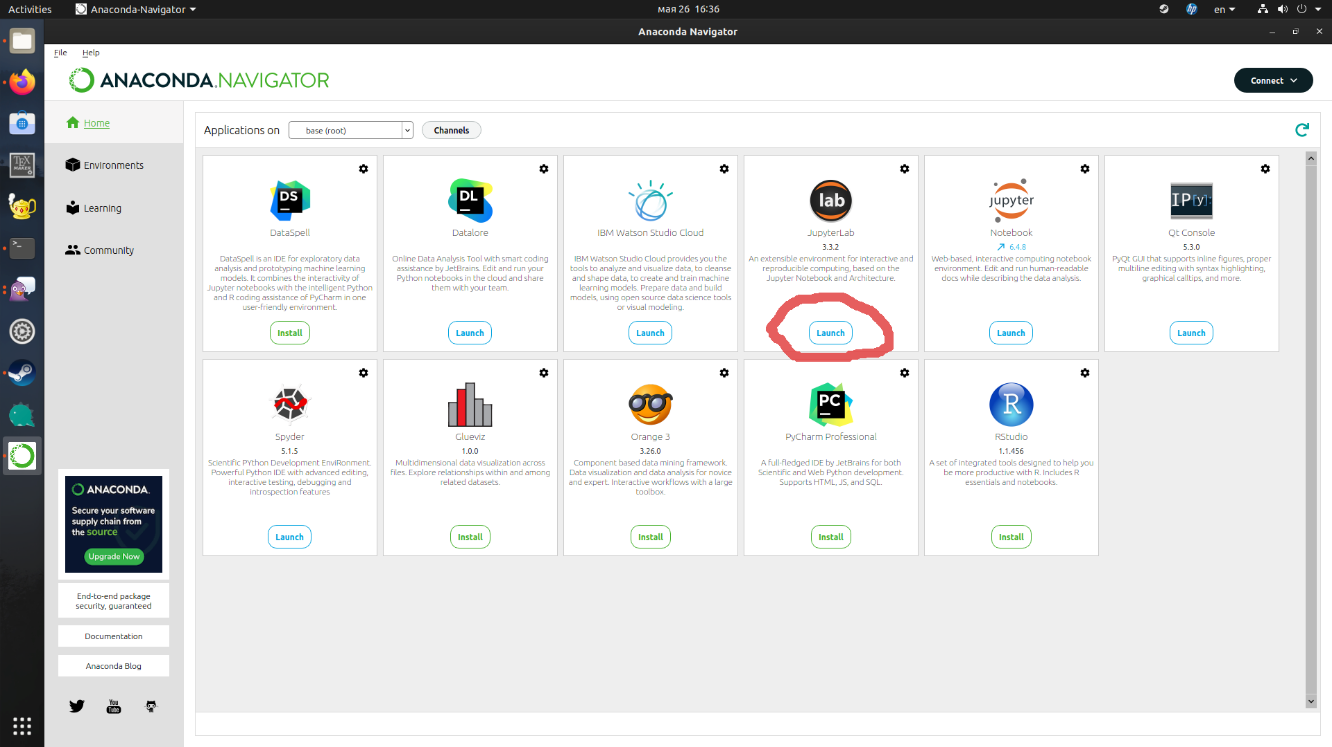
Язык Python – это бесплатно-распространяемый интерпретируемый язык программирования высокого уровня с простым синтаксисом и большим количеством общедоступных дополнительных библиотек, расширяющих его функциональность. Важное замечание: в нашем курсе мы не будет уделять много времени изучению нюансов синтаксиса языка (этому будут посвящены отдельные курсы по программированию), а будем пользоваться Python в первую очередь как инструментом для математических вычислений, познакомившись только с базовыми функциями необходимыми для освоения материала.

Существует несколько способов установки Python на компьютер. Первый способ: скачать дистрибутив с официального сайта [www.python.org](http://www.python.org). Этот дистрибутив содержит минимальный базовый набор библиотек Python и встроенную среду разработки IDLE с базовым функционалом. Главный недостаток такого способа установки в том, что для любой чуть-более серьезной разработки понадобится устанавливать дополнительные библиотеки (для их обозначения часто используют альтернативные термины «пакет» или «модуль»), а также альтернативную среду разработки (более удобную или более функциональную).

Альтернативные варианты установки заключается в установке специализированных сборок (пакетов), содержащих предустановленные дополнительные библиотеки. Наиболее популярный пакет для использования Python в математических расчетах – это пакет Anaconda, который установлен в компьютерном классе (для работа дома его можно бесплатно загрузить с сайта [www.anaconda.com](http://www.anaconda.com)). Этот пакет включает в себя интерпретатор языка Python, набор наиболее часто используемых математических библиотек и удобную интерактивную среду разработки JupyterLab.

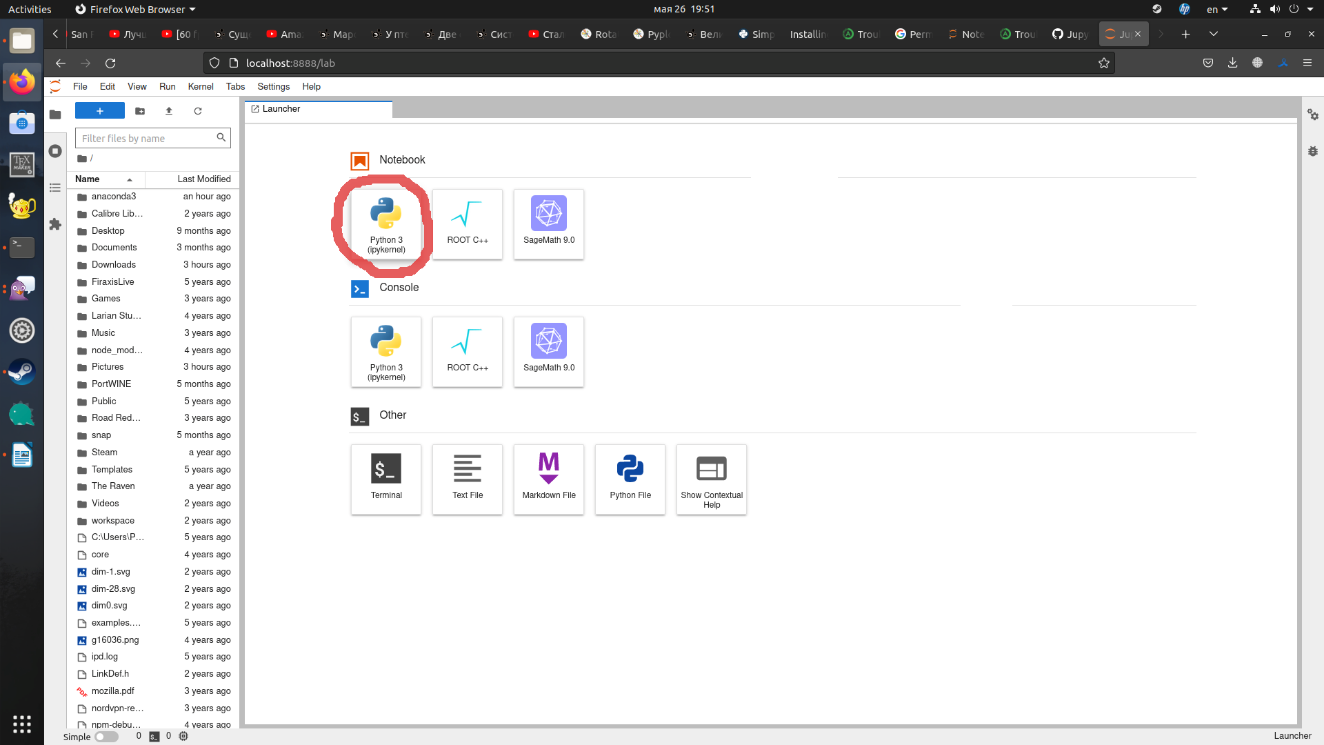
### Работа с окнами пакета Anaconda

После запуска пакета Anaconda загрузится навигатор и предложит на выбор одно из средств разработки (представлено на рисунке 1 ниже, количество и порядок элементов на вкладке может отличаться в зависимости от установленной версии пакета и дополнительных средств разработки).



*Рисунок 1. Навигатор Anaconda*

Для выполнения лабораторных работ по умолчанию будем использовать JupyterLab (на рисунке 1 это четвертый блок в верхнем ряду). После нажатия на кнопку Launch (запуск) в нужном блоке в отдельной вкладке (или новом окне) web-браузера отобразится окно рабочего пространства JupyterLab (рисунок 2).



*Рисунок 2. Стартовое меню JupyterLab*

В левой части стартового окна находится браузер документов со списком файлов в рабочем каталоге программы (по умолчанию – это папка документов пользователя в Windows, либо домашний каталог пользователя в Linux). В правом части окна отображена страница выбора нового документа.

Перед началом работы рекомендуется создать новую подпапку для хранения лабораторных нашего курса. Это можно сделать с помощью кнопки New Folder (кнопка в виде папки с плюсиком) над списком файлов в браузере документов (смотри рисунок 3). Если папка для лабораторных уже была создана (например, на предыдущих занятиях), то перед созданием нового документа надо перейти в эту папку (как в обычном проводнике Windows).

Изображение выглядит как текст

Автоматически созданное описание

*Рисунок 3. Браузер документов.*

Для загрузки файлов с другого диска компьютера в рабочую папку служит кнопка Upload files (кнопка со стрелкой вверх на рисунке 3). Для копирования файла из рабочего каталога на другой диск можно нажать правой кнопкой мыши по имени файла и выбрать пункт меню Download.

### Вычисления в интерактивном окне JupyterLab

Для создания нового интерактивного документа Python в правой части стартового окна JupyterLab находим и нажимаем опцию python 3 (ipkernel) в разделе Notebook (обведено красным кругом на рисунке 2). В результате будет создана новая интерактивная тетрадь Untitled1.ipynb (см. рисунок 4).

Изображение выглядит как текст

Автоматически созданное описание

*Рисунок 4. Рабочая тетрадь JupyterLab.*

*Замечание:* для того, чтобы дать этому файлу более осмысленное имя нажмите правой кнопкой на имени файла в браузере документов и найдите пункт Rename (Переименовать). Также можно воспользоваться главным меню File -> Save Notebook As.

При работе с файлом не забывайте периодически его сохранять с помощью значка с дискеткой.

**Упражнение 0.**

1. В рабочем каталоге JupyterLab создайте отдельную подпапку для лабораторных работ этого практикума.
2. Создайте новый документ с названием, содержащим Вашу фамилию и номер лабораторной (например, Иванов\_lab\_1.ipynb).
3. В этом документе выполнить все последующие упражнения лабораторной.
4. В конце занятия отправить полученный файл на проверку преподавателю.

Команды (код) для вычислений или текст комментариев вводятся в ячейки документа, помеченные символами [ ]. Если надо ввести команду, то на панели инструментов документа надо выбрать «Code». Если надо ввести текст (например, добавить комментарий с номером упражнения), то надо переключить на «Raw» или «Markdown» (во втором случае к тексту можно добавить форматирование с помощью специальной разметки).

Ячейка может содержать как одну команду, так и последовательность из нескольких команд в отдельных строчках.

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 ( \ ).

После ввода команды ее можно выполнить через меню Run или с помощью кнопки в виде черного треугольника (на рисунке 4 обведены). После выполнения команды ее результат отобразится в новой строке после ячейки с этой командой. При этом для последовательности команд отображается только результат последней. Для вывода промежуточных результатов служит специальная команда print().

**Упражнение 1.** Записать и выполнить 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)

5) h=(k+2)\*3+\

3+(k+7)

print(h)

Python поддерживает набор основных математических операций (перечисленных в следующей таблице), что позволяет его использовать для математических вычислений. Однако, для использования более сложных функций, например, квадратного корня или тригонометрических понадобится дополнительных шаг с подключением математической библиотеки, который будет обсуждаться чуть позже в этой работе.

|  |  |  |  |
| --- | --- | --- | --- |
| Сложение | a+b | Модуль числа | abs(a) |
| Вычитание | a-b | Возведение в степень | a\*\*b |
| Умножение | a\*b | Неполное частное | a//b |
| Деление | a/b | Остаток от деления | a%b |

Две последние операции (неполное частное и остаток от деления) являются составляющими операции обобщенного целочисленного деления (или деления с остатком). Деление с остатком a на b означает нахождение таких чисел q и r, что выполняется равенство , где q – целое и . При этом в Python операцию деления с остатком можно применять как к целым числам a и b, так и к вещественным.

**Упражнение 2.** Вычислить значение выражения при значениях переменных .

### Форматы вывода результата вычислений

Вещественные числовые данные, с которыми работает Python, в памяти компьютера хранятся в двоичном виде (как последовательность из 0 и 1). Особенность такого формата хранения в том, что числа кратные ½, ¼, 1/8 (то есть степеням двойки) можно представить конечным числом знаков. Например, число 1.5625=1+1/2+1/16 представляется в виде двоичного числа 1.1001. При этом, например, число 0.3 (с одним знаком после запятой в десятичном виде) в двоичном виде представляется как бесконечная дробь “0.0100110011…”.

При вычислениях компьютер работает только с конечным числом двоичных разрядов, а все «избыточные» знаки отбрасываются. В результате таких вычислений появляются вещественные числа с «мусором» в 15-16-м десятичном разряде.

**Упражнение 3.** Выполните следующие команды:

a = 4.2

b = 2.1

print(a + b)

print(a + b == 6.3)

Для отображения числа на экране без лишнего «мусора» в конце числа, вместо функции печати print используется встроенная функция format.

*Замечание:* функция format не влияет на хранимое в памяти число, а только исправляет его отображение.

*Пример:*

x = 1234.56789

# Два десятичных знака точности после запятой

format(x, '0.2f')

'1234.57'

# Выравнивание по правому краю в 10 символов, один знак точности

format(x, '>10.1f')

' 1234.6'

# Выравнивание по левому краю

format(x, '<10.1f')

'1234.6 '

# Выравнивание по центру

format(x, '^10.1f')

' 1234.6 '

# Включение разделителя разрядов

format(x,',')

'1,234.56789'

format(x, '0,.1f')

'1,234.6'

Если вы хотите использовать экспоненциальную нотацию, измените f на e или E (в зависимости от регистра, который вы хотите использовать для обозначения экспоненты).

format(x, 'e')

'1.234568e+03'

format(x, '0.2E')

'1.23E+03'

**Упражнение 4.** Вычислите значения следующих переменных

и отобразите результата в виде столбца чисел с тремя знаками после запятой и десятичными точками друг над другом.

### Переменные рабочего пространства.

В именах переменных можно использовать только латинские буквы, цифры и символ подчеркивания; большие и малые буквы в именах различаются (X1 и x1 – это разные переменные); имя должно начинаться с буквы (допускается ставить в начале имени знак подчеркивания, но такие имена имеют в Python особый смысл); длина имени не ограничена, но не рекомендуется использовать очень длинные имена; запрещено использовать зарезервированные слова Python и не рекомендуется использовать имена стандартных функций.

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 (h).**

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

**Упражнение 5.**

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.

### Библиотеки Python

Как уже ранее упоминалось, Python по умолчанию дает доступ только к базовым математическим операциям и командам базового синтаксиса. Функциональность Python можно расширить с помощью дополнительных библиотек (в литературе для них также часто используются термины пакет или модуль). Некоторые библиотеки входят в базовую (минимальную) установку Python, часть библиотек устанавливается по умолчанию в составе расширенных пакетов, типа Anaconda, большинство библиотек (особенно узкоспециализированных) приходится устанавливать самостоятельно. На машинах в компьютерном классе все необходимые для лабораторных библиотеки уже установлены.

Библиотек в Python очень много. Далее мы кратко перечислим только те, которые нам понадобятся для математических вычислений.

Библиотека **math** применяется для вычислений с действительными числами и добавляет поддержку основных математических констант (числа e, π) и математических функций (квадратного корня, тригонометрических, показательных и т.д.). Библиотека входит в минимальный пакет установки Python.

Библиотека **cmath** добавляет поддержку операций с комплексными числами. Функции этой библиотеки вы будете подробно изучать в Практикуме по математическому анализу.

Библиотека **mathplotlib** содержит набор методов для создания двумерной графики. В математике обычно используется для построения графиков функций. Подробно работа с этой библиотекой будет рассмотрена в следующих лабораторных.

Библиотека **numpy** предназначена, в первую очередь, для работы с матрицами. Также поддерживает все функции, включенные в библиотеку math.

Библиотека **sympy** предназначена для символьных вычислений. То есть для преобразования формул в буквенном виде и решения задач с помощью символьных формул.

Работа с любой библиотекой в Python начинается с ее подключения (импорта). Существует несколько способов подключения библиотек.

Первый вариант подключение всей библиотеки с помощью команды *import <имя библиотеки>*. После этого можно использовать ее функции с помощью команд вида <*имя библиотеки*>.<*имя функции*>. Например,

import math

print(math.sin(1))

Чтобы каждый раз не писать длинное название библиотеки, можно при подключении задать короткий псевдоним для ее имени.

import numpy as np

print(np.e)

Альтернативный вариант подключения библиотеки – это импорт отдельных или всех ее функций.

from math import sin,pi

После ключевого слова from задается имя библиотеки, которую надо подключить; после ключевого слова import список функций, которые будут использоваться. В дальнейшем коде эти функции можно использовать без указания имени библиотеки.

print(sin(pi/2))

Если требуемых функций много, то допускается импорт всех функций из библиотеки с помощью команды вида:

from numpy import \*

print(log(e))

В этом варианте вместо списка функций стоит звездочка.

*Замечание*: несмотря на удобство использования подключения библиотеки в последней форме записи, почти все руководства по программированию в Python советуют ее избегать. Основная причина в том, что в различных библиотеках могут встречаться функции с одинаковым именем, но при этом разным функционалом. И результат вычислений будет не всегда такой как ожидается.

Например, функция sin есть в библиотеках math, numpy (вычисление синуса в численном виде) и в библиотеке sympy (вычисление в символьном виде). В результате два следующих примера кода будут давать два различных варианта ответа.

from numpy import \*

from sympy import \*

sin(1)

Ответ: sin(1)

from sympy import \*

from numpy import \*

sin(1)

Ответ: 0.8414709848078965

*Замечание:* среди разработчиков на Python нет единого мнения, когда следует подключать необходимые библиотеки. Существуют две основные (противоречащие друг-другу рекомендации): первая рекомендует подключать все библиотеки в начале работы над документом, вторая – подключать только перед первым вызовом функций из библиотеки. Если библиотека или ее функция уже была подключена, то подключать ее повторно в каждой ячейке не надо.

**Упражнение 6.**

Вывести на печать значения математических констант е и π.

### Математические операции и функции в Python и его библиотеках

Математические выражения в Python, как Вы уже видели ранее, используют обычные арифметические операции: сложение (+), вычитание (-), умножение (\*), деление ( / ), возведение в степень (\*\*). Порядок вычислений определяется правилами старшинства как в обычной арифметике. Обратите внимание, что знак операции умножения между скобками и переменными опускать нельзя, как это допустимо в обычной математике.

Основные математические функции можно подключить либо с помощью библиотеки math, либо библиотеки numpy.

Основные элементарные функции, которые могут потребоваться при выполнении лабораторных работ, приведены в таблице ниже. Обратите внимание, что название некоторых функций может отличаться от привычных нам по математическим формулам (это связано с тем, что при наименовании функций используются западные стандарты). Более полный список функций библиотек можно посмотреть в их документации.

|  |  |  |  |
| --- | --- | --- | --- |
| **Функция** | **Команда** | **Функция** | **Команда** |
| (степень) | x\*\*n  pow(x,n) | (модуль) | abs(x) |
|  | sqrt(x) |  | exp(x) |
|  |  |  | expm1(x) |
| ln x | log(x) |  | log10(x) |
|  | log2(x) |  | log(x, a) |
| cos (x) | cos(x) | sin (x) | sin(x) |
| tg (x) | tan(x) | ctg (x) | cot(x) |
| arccos (x) | acos(x) | arcsin (x) | asin(x) |
| arctg (x) | atan(x) | arcctg (x) | acot(x) |
| ch (x) | cosh(x) | sh (x) | sinh(x) |

Аргумент всех функций обязательно заключать в круглые скобки, в противном случае это считается ошибкой.

Обратите отдельное внимание на использование функции экспонента () и функцию pow. В отличии от обычной математики, в которой эти функции рассматриваются как операции возведения константы е или числа в степень, в Python используется синтаксис, аналогичный тригонометрическим или логарифму. Например, выражение будет иметь вид exp(2\*x), а выражение записывается как pow(pi,3). Также для этих двух функций можно использовать знак операции возведения в степень (\*\*).

**Упражнение 7.**

Вычислить значения следующих выражений:

,