

Начало работы с SageMath

Eugene Strakhov

Please report bugs to strakhov.e.m@onu.edu.ua

1 Что такое SageMath?

SageMath — открытое и свободное программное обеспечение для различного рода математических расчётов, задуманное как альтернатива коммерческим пакетам MATLAB, Maple, Mathematica и др. Проект основан Уильямом Стайном (William Stein), математиком из Университета Вашингтона. Первая версия **SageMath** вышла 24 февраля 2005 года.

Первоначально система называлась **SageMath** (System for Algebra and Geometry Experimentation), которое затем трансформировалось в **SageMath** (*англ.* «мудрец», «шалфей»). После ребрендинга 2016 года и основания компании SageMath, Inc. система называется **SageMath**.

Аналитические возможности **SageMath** охватывают такие области математики, как алгебра, теория чисел, комбинаторика, анализ, численные методы и др. Достоинствами **SageMath** являются обширная справочная система, качественная 2D и 3D графика, открытый исходный код и поддержка со стороны пользователей. Значительная часть **SageMath** написана на языке Python. Этот же язык используется в качестве основного языка синтаксиса. По умолчанию в **SageMath** включены все компоненты стандартной библиотеки Python.

SageMath ориентирована на всех, кто интересуется математическими расчётами на компьютере, — от школьников и студентов до профессоров и научных работников.

2 Что такое CoCalc?

CoCalc (Collaborative Calculation in the Cloud) — облачный сервис, построенный по принципу «всё включено». Создав аккаунт, вы получаете доступ к большому количе-

ству продуктов, включающих ПО для математических расчётов (SageMath, Octave, Julia и др.), программирования (Python 2.X, Python 3.X, Java, Perl, Ruby), анализа и визуализации данных (R), а также создания научной и технической документации (L^AT_EX, Markdown, HTML). В облаке CoCalc имеется возможность создания и работы с Jupyter Notebook — интерактивной веб-оболочкой IPython, позволяющей объединять собственно код, текстовые комментарии, формулы, виджеты, графики, диаграммы и являющейся незаменимым инструментом Data Scientist'a. Кроме того, CoCalc обладает удобным функционалом для поддержки учебных курсов (добавление учащихся, выдача и приём заданий, рассылка учебных материалов, чат) и для совместной работы над проектами (наподобие GitHub). В последнее время в проект добавлена обширная библиотека тьюториалов, онлайн-учебников, L^AT_EX-шаблонов и др. по тематике научных вычислений. Все возможности CoCalc предоставляются пользователям в готовом к использованию виде и совершенно бесплатно. Возможна также платная подписка (начиная с 7\$ в месяц) с целью улучшения скорости соединения и получения доступа к большему количеству вычислительных мощностей.

3 Начало работы

Далее предполагается, что у вас имеется аккаунт CoCalc и создан проект.

SageMath поддерживает интерфейс рабочих листов. Рабочий лист состоит из ячеек, каждая из которых может быть независимым скриптом. Интерфейс является интерактивным, т. е. вы имеете возможность сразу же увидеть результат работы вашего скрипта. Для запуска ячейки установите курсор на неё и нажмите комбинацию клавиш **Shift** + **Enter** либо зелёную кнопку **Run** на панели инструментов.

Рабочие листы сохраняются на сервере автоматически примерно каждые 45 секунд. Для принудительного сохранения нажмите зелёную кнопку **Save** на панели инструментов. **Внимание! Если кнопка имеет ярко-зелёную окраску, это означает, что документ НЕ сохранён. Светло-зелёная окраска — документ сохранён.**

Вы можете восстанавливать предыдущие версии вашего документа с помощью голубой кнопки **TimeTravel** на панели инструментов.

Если имеются неполадки в работе с системой (текст удаляется/появляется сам собой, программа «висит» и т. п.), вы можете остановить выполнение скрипта (кнопка **Stop**) либо перезапустить рабочий лист (кнопка **Restart**). **Внимание! Нажимая «Restart», вы сбрасываете все вычисления, которые производились до этого. Их результаты всё ещё будут отображаться, однако значения переменных будут недоступны.**

Чтобы перезапустить весь проект, зайдите в меню `Settings` (отображается рядом с именами открытых файлов).

В правом верхнем углу панели инструментов есть кнопка «Чат». Чат открывается справа от рабочего листа. В чате поддерживается синтаксис Markdown, L^AT_EX-формулы и др. Вы также можете редактировать свои сообщения, созданные ранее. Уведомления о новых сообщениях в чате отображаются в верхнем меню, как на Facebook.

4 Особенности синтаксиса

Базовым языком программирования в SageMath является Python 2.X. Выделим несколько важных особенностей синтаксиса:

- В SageMath операции `^` и `**` являются синонимами и обозначают возведение в степень.
- Дроби вида `a/b`, где `a` и `b` — целые числа, являются **символьными константами** и не поддерживают автоматическое приведение к типу `float`. См. примеры ниже.
- Дроби вида `a.0/b` и `a/b.0`, где `a` и `b` — целые числа, автоматически приводятся к типу `float`.
- Чтобы вывести объект на экран, достаточно просто написать его имя (для выражений — достаточно написать само выражение), как при работе в режиме интерпретатора в IDLE. См. примеры ниже.
- `print` в языке Python 2.X является командой, а не функцией, поэтому круглые скобки для аргументов не обязательны. См. примеры ниже.
- Для вывода строк Юникода при помощи `print` необходимо использовать префикс `'u'`.

```

2^5 == 2**5 # Возведение в степень: ^ и ** эквивалентны
print 1/2 # 1/2 - символьная константа
print 1/2-1/3 # с дробями можно производить вычисления
a = 1/3.0 # значение типа float
b = 1.0/3 # значение типа float
print a, b # скобки можно не ставить
print a, u'это то же самое, что', b # обратите внимание на префикс 'u'

True
1/2
1/6
0.3333333333333333 0.3333333333333333
0.3333333333333333 это то же самое, что 0.3333333333333333

```

5 Математические константы и функции

В SageMath по умолчанию содержатся все компоненты стандартной библиотеки Python, так что вам не нужно вручную подключать модули `math`, `random` и другие. Все функции и константы, описанные в модуле `math`, доступны по именам. Разумеется, синтаксис SageMath чувствителен к регистру, так что `Pi` и `pi` — не одно и то же. Встроенные имена констант, такие как `pi`, `e` переопределять не рекомендуется. Заметьте также, что значения `pi`, `e` по умолчанию не приводятся к типу `float`, как в «чистом» Python. Здесь это символьные константы.

```

sin(pi/4) # вычисляется в символьном виде
ln(e) # ln() - синоним для log()
ln(2) # символьная константа
arctan(1) # вычисляется в символьном виде
arctan(1.0) # вычисляется в виде float-числа

1/2*sqrt(2)
1
log(2)
1/4*pi
0.785398163397448

```

6 Функция `numerical_approx()`

Чтобы вычислить значение выражения в виде числа (как говорят математики, «довести ответ до числа»), можно явно привести его к типу `float`, например:

```
float(sin(pi/4))
```

что в результате даст значение 0.707106781187. Однако для этой цели лучше подойдёт функция (или метод) `numerical_approx()`, позволяющий дополнительно указывать точность представления числа. «Длинное и плохо запоминающееся название, совсем не в духе Python», — можете заметить вы. К счастью, для многих функций и методов с длинными названиями в **SageMath** предусмотрены псевдонимы. Для рассматриваемой нами функции это `n()`. Оба названия эквивалентны.

```

a = sin(pi/4)
# n() как метод
a.numerical_approx()
a.n()
# n() как функция
n(a)
numerical_approx(a)
# Дополнительные аргументы:
# prec - точность в битах,
# digits - количество значащих цифр
n(a, prec=10)
n(a, digits=40)
# и даже так!
n(pi, digits=1000)

```

```

0.707106781186548
0.707106781186548
0.707106781186548
0.707106781186548
0.71
0.7071067811865475244008443621048490392848
3.1415926535897932384626433832795028841971693993751058209749445923
078164062862089986280348253421170679821480865132823066470938446095
505822317253594081284811174502841027019385211055596446229489549303
819644288109756659334461284756482337867831652712019091456485669234
603486104543266482133936072602491412737245870066063155881748815209
209628292540917153643678925903600113305305488204665213841469519415
116094330572703657595919530921861173819326117931051185480744623799
627495673518857527248912279381830119491298336733624406566430860213
949463952247371907021798609437027705392171762931767523846748184676
694051320005681271452635608277857713427577896091736371787214684409
...

```

7 Вызов справки

Получить справку по какой-либо функции можно, указав её имя и знак вопроса. Справка, как правило, сопровождается примерами. Двойной знак вопроса выводит исходный код функции.

```
# Тестируем справку: функция n()
```

```
n?
```

```
...
```

```
n??
```

```
...
```

8 `range()`, `xrange()`, `srange()` и `xrange()`

Как вы помните, в Python 3.X функция `range()` генерировала т. н. «виртуальную последовательность», т. е. ряд значений, представляющий собой арифметическую прогрессию, при этом генерируемые значения не хранились в памяти.

В Python 2.X «обязанности» `range()` выполняет функция `xrange()`, а просто `range()` создаёт **список** из тех же значений и потому работает слегка медленнее. Если для вас это не принципиально, вы вольны использовать любую из этих функций. Заметим, что в Python 3.X функция `xrange()` отсутствует за ненадобностью.

Помимо вышеуказанных функций, в SageMath поддерживаются также последовательности с произвольным (а не только целочисленным) шагом. Это `srange()` и `xrange()` (префикс 's' обозначает SageMath). Разница между ними ровно та же, что и между `range()` и `xrange()` в Python 2.X. Если быстродействие вас не очень заботит, пользуйтесь любой из них.

Последовательности `srange()` и `xrange()`, кроме нецелочисленных шагов, также позволяют включать в рассмотрение конец промежутка (опция `include_endpoint`).

```

range(10) # здесь range - это сразу список!
tuple(srange(10)) # srange() можно привести к различным типам
srange(1, 10, 1/2)
xrange(0, 2*pi, pi/2) # Опа! Ничего не вышло
list(xrange(0, 2*pi, pi/2)) # другое дело
# Необязательный аргумент include_endpoint=True (включить конечную точку)
srange(1, 10, 1/2, include_endpoint=True)

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
[1, 3/2, 2, 5/2, 3, 7/2, 4, 9/2, 5, 11/2, 6, 13/2, 7, 15/2, 8, 17/2, 9, 19/2]
<generator object at 0x7f2dae7e7ef0>
[0, 1/2*pi, pi, 3/2*pi]
[1, 3/2, 2, 5/2, 3, 7/2, 4, 9/2, 5, 11/2, 6, 13/2, 7, 15/2, 8, 17/2, 9, 19/2, 10]
156

```

9 Режим %md (Markdown)

Если в начале ячейки поставить команду %md, дальнейшее содержимое будет интерпретироваться как Markdown-код. Изменится также и вид панели инструментов. Markdown — специализированный упрощённый язык разметки, совместимый с более продвинутыми языками (в частности, HTML). С помощью Markdown вы можете включать в ваш рабочий лист форматированный текст с несколькими уровнями заголовков, а также изображения, ссылки, таблицы, формулы и др.

10 Режим %latex

Если в начале ячейки стоит команда %latex, её содержимое будет обработано как ЛАТЭХ-код. По умолчанию кириллица в нём работать не будет, однако если перед включением режима %latex прописать необходимые пакеты, всё заработает. Заметим, что нижеследующие строки необходимо скомпилировать в режиме %sage (т. е. режиме, выставленном по умолчанию):

```

latex.add_to_preamble("\\usepackage[T2A]{fontenc}")
latex.add_to_preamble("\\usepackage[utf8]{inputenc}")
latex.add_to_preamble("\\usepackage[english, russian]{babel}")

```

Обратите также внимание на кавычки и двойные обратные слэши.

Прописав нужное количество пакетов, вы сможете «заставить» ваш рабочий лист

делать все те *удивительные вещи*, на которые способен \TeX .

См. также <http://doc.sagemath.org/html/en/reference/misc/sage/misc/latex.html>

11 Вывод с помощью `show()`

Кроме `print`, поддерживается другая команда для вывода: `show()`. Особенность её в том, что она без дополнительных указаний справляется со строками Юникода, а также улучшает внешний вид формул (в частности, дробей). Содержимое команды `show()` выводится по центру.

```
# Тестируем функцию show()
show('Привет, мир!')
L = [pi/n for n in range(1, 10)]
show('Это список:', L)
show('Число', pi^e, 'примерно равно', (pi^e).n())
```

Привет, мир!
 Это список: $\left[\pi, \frac{1}{2}\pi, \frac{1}{3}\pi, \frac{1}{4}\pi, \frac{1}{5}\pi, \frac{1}{6}\pi, \frac{1}{7}\pi, \frac{1}{8}\pi, \frac{1}{9}\pi\right]$
 Число π^e примерно равно 22.4591577183610

12 Вывод с помощью `latex()`

Команда `latex()` возвращает строку специального вида — \LaTeX -представление содержимого вывода. Может использоваться в комбинации с `show()`. Генерирует код, готовый для вставки в \LaTeX -документ.

```
# Тестируем функцию latex()
latex(sin(pi/4))
latex([pi/n for n in range(1, 10)])
# Использование в комбинации с show()
show('$\sin\frac{\pi}{4}='+latex(sin(pi/4))+'$')
# Забежим немного вперёд...
a, b = var('a b')
expr = (a+b)^2
show('$'+latex(expr)+'='+latex(expr.expand())+'$')
```


$\frac{1}{2} \setminus, \sqrt{2}$

$\left[\pi, \frac{1}{2} \setminus, \pi, \frac{1}{3} \setminus, \pi, \frac{1}{4} \setminus, \pi, \frac{1}{5} \setminus, \pi, \frac{1}{6} \setminus, \pi, \frac{1}{7} \setminus, \pi, \frac{1}{8} \setminus, \pi, \frac{1}{9} \setminus, \pi\right]$


$$\sin \frac{\pi}{4} = \frac{1}{2} \sqrt{2}$$

$$(a+b)^2 = a^2 + 2ab + b^2$$

13 Экспорт рабочего листа в HTML

Рабочий лист SageMath может быть сконвертирован в формат HTML за несколько мгновений. Для этого нажмите иконку  на панели инструментов (она выглядит как принтер) — готово!

14 Экспорт рабочего листа в PDF (via L^AT_EX)

Ваш рабочий лист также может быть успешно сконвертирован в формат PDF. Для этого нажмите иконку  рядом с «принтером», а затем (настоятельно рекомендуется!) активируйте опцию «Keep generated files...» в открывшемся окне. Здесь, однако, придётся слегка «повозиться» с исходным кодом. Файл `.tex` будет сгенерирован автоматически, а далее в нём нужно «навести порядок», чтобы корректно отобразить кириллицу.