

Nbgrader.

Винецкая Полина

15 августа 2020 г.

Содержание

1	Установка. Для работы с сервером не нужно	2
2	Создание курса	2
2.1	Шаблон	2
2.2	Создание своего курса	2
3	Структура	3
3.1	Команды	3
3.2	Вид	3
3.2.1	Учитель	3
3.2.2	Студент	4
4	Работа с nbgrader	4
4.1	Formgrader	4
4.2	Создание задания	5
4.2.1	С помощью Formgrader	5
4.2.2	Командная строка	5
4.3	Добавление тестов	5
4.3.1	Read only	5
4.3.2	Manually graded task	6
4.3.3	Manually graded answer	6
4.3.4	Autograded answer	6
4.3.5	Autograder tests	6
4.4	Проверка	7
4.4.1	Formgrader	7
4.4.2	Командная строка	7
4.5	Generate	7
4.5.1	Formgrader	7
4.5.2	Командная строка	7
4.6	Публикация	8
4.6.1	Папка exchange	8
4.6.2	Formgrader	8
4.6.3	Командная строка	8
4.7	Сбор	8
4.7.1	Formgrader	9
4.7.2	Командная строка	9
4.8	Проверка	9
4.8.1	Автопроверка	9

4.8.2 Вручную	9
4.9 Feedback	9
5 Работа с базой данных	10
5.1 Студенты	10
5.2 nbgrader_config.py	10
5.2.1 Скрипт	10
5.2.2 Командная строка	10
6 Random useful things	10

1 Установка. Для работы с сервером не нужно

Для установки nbgrader откройте терминал и введите команду:

```
pip install nbgrader
```

или:

-

```
conda install jupyter
```

```
conda install -c conda-forge nbgrader,
```

если вы используете анаконду.

Для получения доступа к возможностям учителя, необходимо установить расширения:

```
jupyter nbextension install --sys-prefix --py nbgrader --overwrite
```

```
jupyter nbextension enable --sys-prefix --py nbgrader
```

```
jupyter serverextension enable --sys-prefix --py nbgrader
```

Первое расширение необходимо для создания заданий. Для assignment list, formgrader, и validate необходимы оба расширения.

2 Создание курса

Есть два пути: создать свой курс с нуля или использовать шаблон.

2.1 Шаблон

Команда

```
nbgrader quickstart course_id
```

создаст курс с названием course_id и наполнением из 2 заданий и 2 студентов. На примере шаблона проще всего разобраться в работе nbgrader'a

2.2 Создание своего курса

Создается папка со структурой, описанной в п.3, файлом nbgrader_config.py и базой данных. Мне кажется, удобнее редактировать шаблон, т.к. повторить вручную всю структуру курса довольно сложно.

3 Структура

3.1 Команды

Команды nbgrader строятся следующим образом:

```
{course_directory}/{nbgrader_step}/{student_id}/{assignment_id}/{notebook_id}.ipynb
```

course_directory – корневая папка, в которой хранится все (информация о классе, заданиях и т.д.)

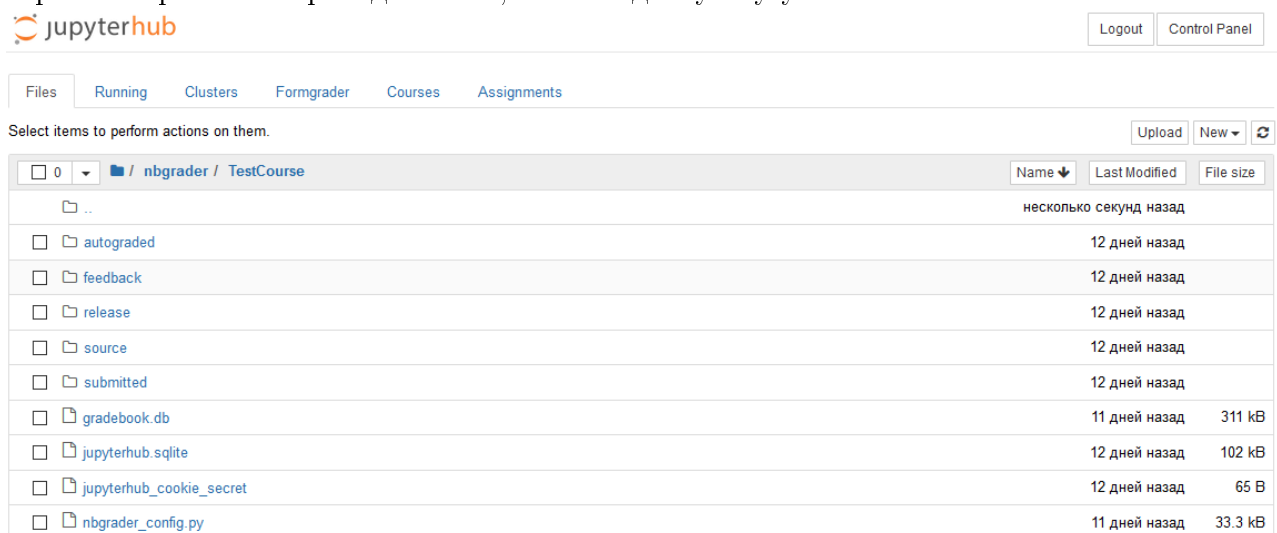
nbgrader_step – команда (проверить, забрать, раздать задания и т.д.). Есть тонкости с тем, где запускать команды и где будет видно их результат. Далее подробнее.

*_id можно заменить звездочкой, если нужно выполнить команду для всех элементов.

3.2 Вид

3.2.1 Учитель

Структура nbgrader'a устроена так: есть две папки: корневая (couse_directory) и обменная (exchange). Корневая папка содержит в себе информацию о курсе, учениках, заданиях. В ней хранятся решения преподавателя, и к ней доступа у учеников нет.



Обменная папка создана для обмена между учениками и преподавателем, в ней хранятся: а) Выложенные задания учителя без решений б) Решения студентов в) Фидбэки учителя и nbgrader на каждое задание каждого ученика. При работе с сервером напрямую в нее доступа ни у кого нет.

Для работы без сервера Для работы со сторонним облаком вроде Dropbox студенты и учитель должны поставить nbgrader локально. "Студенческая" версия ставится в две строчки и требует только анаконду и питон версии не ниже 3 (раздел 7). Можно установить на почти любую операционную систему, но как показала практика, не на 32-битную.

Основная проблема, с которой столкнется преподаватель, если у учеников не стоит nbgrader, это безумные таймстемпы, без которых nbgrader не будет проверять задания и отсылать фидбэки. Пока я не нашла способа отключить их, поэтому, если ученики не ставят себе nbgrader, преподавателю придется таймстемпы генерировать с помощью какого-то скрипта.

Что есть в папке курса gradebook.db – База данных типа sQlite database, находится в

`{course_directory}/gradebook.db`.

Внутри нее лезть обычно необходимости нет, nbgrader все манипуляции выполняет сам.

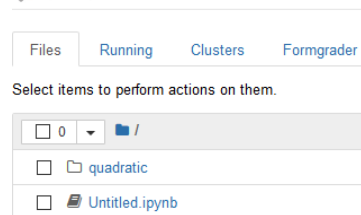
`nbgrader_config.py` – также находится в корневой папке курса. Хранит всю информацию о курсе, связанную с базой данных(адрес обменной папки, информация об учениках и т.д)

Например,

`c.CourseDirectory.course_id = "course101" -- метод, задающий id курса`

3.2.2 Студент

У студента тоже есть корневая папка, соответствующая курсу (можно подписаться на несколько разных курсов, тогда и корневых папок будет несколько). Однако в ней отобра-



жаются только те задания, на которые студент подписался (функция fetch).

4 Работа с nbgrader

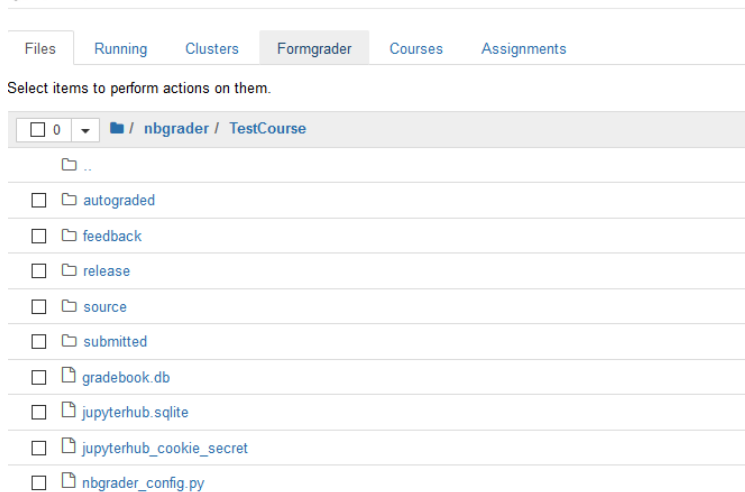
Все, необходимое учителю, укладывается в схеме

`Make assignment -- Generate -- Release -- Collect -- Grade -- Feedback`

4.1 Formgrader

Расширение, позволяющее работать не через командную строку, а прямо в Jupyter.

Если все расширения из пункта 1 были установлены верно(если вы устанавливаете



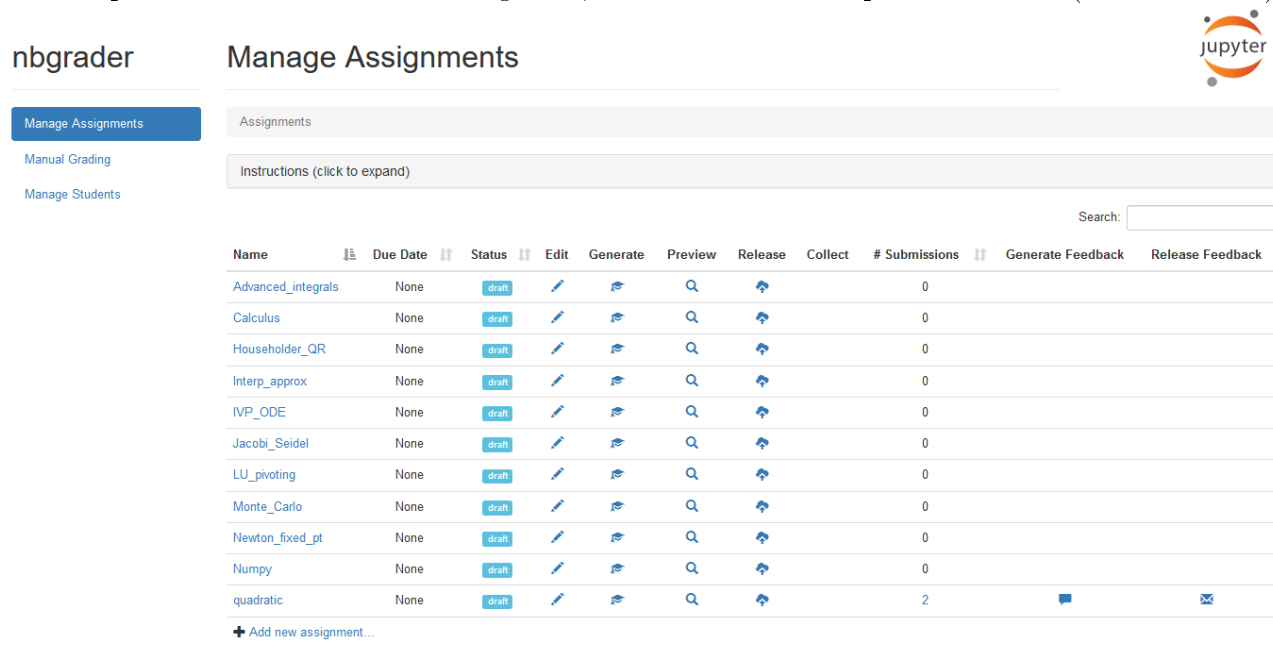
nbgrader локально), в Jupyter появится вкладка:

Возможная проблема (не для сервера) Если вы видите табличку “The course id has not been set in nbgrader_config.py“, а в графе formgrader “No data available in table“, значит, вы запустили Jupyter находясь не в корневой папке курса.

4.2 Создание задания

4.2.1 С помощью Formgrader

Здесь все просто: нажимаете +add assigment, добавляете имя и время дедлайна(опционально).



Name	Due Date	Status	Edit	Generate	Preview	Release	Collect	# Submissions	Generate Feedback	Release Feedback
Advanced_integrals	None	draft						0		
Calculus	None	draft						0		
Householder_QR	None	draft						0		
Interp_approx	None	draft						0		
IVP_ODE	None	draft						0		
Jacobi_Seidel	None	draft						0		
LU_pivoting	None	draft						0		
Monte_Carlo	None	draft						0		
Newton_fixed_pt	None	draft						0		
Numpy	None	draft						0		
quadratic	None	draft						2		

В таблице formgrader появится имя задания, в эту папку нужно поместить ноутбуки с задачами (их может быть несколько) (Зайти в папку можно кликнув на имя в таблице)

4.2.2 Командная строка

Все задания, созданные вами, но еще не опубликованные, хранятся в папке курса в разделе source, поэтому для добавления задания с названием assignment_id можно просто создать папку course_id/source/assignment_id и в нее поместить задачи, однако тогда надо обновить соответствующую строку в nbgrader_config.py.

4.3 Добавление тестов

Для того, чтобы сделать задачи удобоваримыми для nbgrader, нужно добавить тесты и ответы к ним.

Важно Nbgrader прячет от студентов решения и тесты, если использовать соответствующие brackets. Все brackets и их содержимое будут скрыты от студента и заменены на фразы типа “Введите решение“. В ноутбуке с заданием нужно нажать View – Cell Toolbar: откроются все возможности nbgrader.

Нажмите Create Assignment. В правом углу каждой ячейки появится окошко с выбором вида этой ячейки, а после выбора появится id этой ячейки (по этим id nbgrader будет ориентироваться при проверке и прятать нужные ячейки от студентов при релизе). Слева от id появится ячейка с количеством баллов за задание. Ячейки, подсвечивающиеся синим, не будут доступны студентам в своем первоначальном виде.

4.3.1 Read only

Обычный комментарий. Отличается от ячейки markdown тем, что является неизменяемым для студента. В formgrader это отмечено знаком замка в левом углу ячейки.

Важно В зависимости от версии nbgrader, функционал таких ячеек изменяется. На текущей версии, если я правильно поняла, все работает так:

При генерации задания содержимое ячейки записывается в базу данных. Во время проверки nbgrader заменяет содержимое этой ячейки в файле студента на то, что было сохранено в базе данных на момент релиза. То есть например, вариант удалить ячейку с тестами и получить 100% не прокатит, nbgrader восстановит ячейку и проверит код. Именно поэтому тесты и решения пишутся в разных ячейках: тесты также обладают функциями ячейки Read-only. (об этом в 4.2.5)

4.3.2 Manually graded task

Задания, требующие индивидуальной проверки учителем каждого ученика и нескольких ячеек, например “приведите графики, придумайте тесты, хорошо показывающие недостатки модели“. В самой ячейке вводится только текст задания, ученик (наверное?) создаст новые ячейки, в которых выполнит задание. Для этого типа ячеек можно добавить решение в brackets `=== BEGIN MARK SCHEME === ... === END MARK SCHEME ===`, например, для облегчения работы ассистентам. Эта часть будет скрыта от студента.

4.3.3 Manually graded answer

Задания, требующие индивидуальной проверки учителем каждого ученика, например “объясните эффект“или “напишите формулу“. Выполняются в той же ячейке.

Важно Сам текст задания нужно писать в другой ячейке, вида Read only, так как содержимое Manually graded answer ячеек nbgrader заменит на фразу YOUR ANSWER HERE в версии студента.

4.3.4 Autograded answer

Доступны только для ячеек с кодом. В brackets `### BEGIN SOLUTION ... ### END SOLUTION` нужно написать решение. Если написать решение, но не использовать brackets, nbgrader скроет от студента все содержимое ячейки, но исключения не выдаст. Баллы за эту ячейку не выставятся, так как критерием являются тесты.

4.3.5 Autograder tests

Доступны только для ячеек с кодом. Необходимо использовать функции типа `numpy.allclose` или похожие. Если все тесты проходят, студент получает число баллов в ячейке. Хотя бы 1 тест падает – 0 баллов за задание. Если вы хотите чтобы тесты были скрыты от студента, можно использовать brackets `### BEGIN HIDDEN TESTS` и `### END HIDDEN TESTS`.

Важно Эта функция Особенно важна для тестирующих ячеек, которые всегда помечены как read-only. Из-за того что механизм автопроверки выставляет студенту максимальный балл в случае прохождения всех тестов, простым способом обойти это было бы просто удалить или закомментировать тесты. Поэтому ячейки read-only игнорируют все изменения, сделанные студентом. А если работать через сервер, эти ячейки будут просто не кликабельны.

Очень важно Если изменить количество ячеек при работе (на сервере так сделать невозможно), залить задания получится, а проверить нет. Nbgrader каждой ячейке дает индивидуальный id, и если хоть бы одну из ожидаемых ячеек не находит, работать отказывается.

4.4 Проверка

4.4.1 Formgrader

Чтобы проверить адекватность тестов, достаточно нажать кнопку Validate на панели Jupyter сверху. Nbgrader запустит ячейки, и в случае, если ваше решение успешно обрабатывает все тесты, выдаст: “Success! Your notebook passes all the tests.”

В противном случае вы увидите на экране “обвалившиеся” ячейки.

4.4.2 Командная строка

Для командной строки существует команда `nbgrader validate assignment_id`. Однако чтобы проверить нужный ноутбук, нужно указать его адрес относительно корневой папки курса.

```
nbgrader validate source/assignment_id/problem_id.ipynb
```

4.5 Generate

Создает версию для релиза (ту, которую увидит студент), убирая ответы и скрытые тесты из задания, лежащего в папке source. Однако студенту сгенерированные задания все еще будут недоступны.

4.5.1 Formgrader

Чтобы сгенерировать задание, надо перейти во вкладку formgrader и нажать на значок шапки в графе generate напротив соответствующего задания. Вы увидите окно log output.

Возможная проблема В папке source должен лежать файл `header.ipynb`; если его там нет, его нужно скачать и вставить. По смыслу это шапка, в которой студент напишет информацию о себе, без нее будет выдаваться странная ошибка. Для того чтобы увидеть версию для релиза, можно нажать на значок лупы в строчке с соответствующим заданием во вкладке formgrader.

4.5.2 Командная строка

Для командной строки существует функция `nbgrader generate assignment_id`. Перед использованием команды убедитесь, что нужный ноутбук лежит по адресу

```
{course_directory}/source/{assignment_id}/{notebook_id}.ipynb
```

Сгенерированное задание будет лежать по адресу

```
{course_directory}/release/{assignment_id}/{notebook_id}.ipynb
```

В результате папки source и release будут аналогичны по структуре, только в первой будут лежать ноутбуки преподавателя, с решениями и тестами, а во второй – те же файлы, но зацензурированные, для учеников.

Если вы хотите проверить через командную строку, что ваши действительно спрятаны в версии для студента, можно использовать команду

```
nbgrader validate --invert release/{assignment_id}/{notebook_id}.ipynb
```

Если все хорошо, то есть файл на месте, и все тесты обвалились, вывод будет следующим: “Success! The notebook does not pass any tests”

4.6 Публикация

При работе локально релиз происходит так: по адресу(про него в 4.5.0) nbgrader создает папку с названием курса(папку для обмена), а в ней две директории: outbond для релиза и inbond для сбора(о ней в разделе 4.6).

То есть результат действия “публикация задания” – это перемещение ноутбуков, соответствующих этому заданию в папку outbond из папки course_directory/release. Папка outbond по организации и содержанию выглядит точно так же, как и release в папке курса, однако находится не в папке курса. (По факту папка курса должна быть доступна только учителю, а outbond расшарена ученикам).

Для работы с сервером таких проблем нет.

4.6.1 Папка exchange

В файле get_config.py можно вписать адрес папки для обмена между студентами и преподавателем:

```
c = get_config()

c.CourseDirectory.course_id = "example_course"
c.Exchange.root = "/tmp/exchange" например
```

В последней строке вписываем путь к папке обмена, которая уже должна существовать и все пользователи должны иметь право изменять/удалять/создавать файлы. Кнопка release во вкладке formgrader перемещает ноутбуки по этому адресу. Если ничего не указывать, адрес папки обмена:

```
/srv/nbgrader/exchange/course_id/outbound/
```

4.6.2 Formgrader

Тут снова все просто: в вкладке formgrader нужно нажать на значок в виде облака для релиза, и на крестик для того, чтобы релиз отменить.

Возможная проблема Если вы поменяли адрес папки обмена, но релиз произошел по старому адресу, перезапустите ядро Jupyter.

4.6.3 Командная строка

Команда nbgrader release assignment_id выполняет релиз по адресу, указанному в файле get_config.py.

4.7 Сбор

(если у студента не установлен nbgrader, а сервер не используется) Такого не предполагает функционал, поэтому часть автоматических действий перекладывается на учителя.

В папке `{exchange_directory}/{course_id}/inbound/` должны лежать папки студентов со структурой `{student_id+assignment_id+timestamp}/.ipynb`

Пример:

```
/tmp/exchange/TestCourse/inbound/vivanov+quadratic+2020-04-18 17:50:45.884871 UTC/  
/quadratic_exercise.ipynb
```

Проблема Я пока не нашла способа избежать такой сложной структуры, так как она создается nbgrader'ом, когда студенты отсылают свои решения. Возможно он есть. Если изменить имя папки, функция collect работать не будет.

4.7.1 Formgrader

во вкладке formgrader в столбике `#submissions` будет указано количество решений. Для того, чтобы их “забрать”, надо нажать на облачко в графе collect.

4.7.2 Командная строка

Функция nbgrader collect

4.8 Проверка

4.8.1 Автопроверка

Для автопроверки нужно нажать на количество сданных решений, а затем на значок молнии. nbgrader выставит баллы за каждое автоматически проверенное задание и изменит статус ассигмента.

4.8.2 Вручную

Если есть задания, требующие проверки вручную, вам об этом напомним статус задания. Чтобы выполнить проверку, нужно выбрать вкладку Manual Grading слева в форме formgrader. Далее нажать на задание, затем на ноутбук, и щелкнуть на первую работу из сданных. Откроется окно, в котором около каждого из ячеек можно выбрать количество баллов, в том числе изменить оценку, выставленную nbgrader автоматически. Сверху есть две вкладки `← Prev` и `Next →`, с помощью которых можно переключаться между ноутбуками учеников.

4.9 Feedback

С формированием фидбэка все просто: после проверки задания во вкладке formgrader нужно нажать на значок письма во вкладке Generate Feedback. В корневой папке, по адресу `/feedback/{student_id}/{assignment_id}/` будет лежать фидбэк в формате .html. А вот с отправкой фидбэка есть проблемы. Из-за отсутствия таймстэмпов(см) nbgrader отказывается перемещать файлы в обменную папку. Пока что есть два варианта решения: либо руками копировать папку фидбэк в обменную папку (а точнее `exchange/{course_id}/feedback`), либо создавать таймстэмпы вручную, но я пока не знаю, как)

5 Работа с базой данных

5.1 Студенты

Проверить список студентов можно с помощью команды `nbgrader db student list` в командной строке. Существует несколько способов добавить в класс студента:

5.2 `nbgrader_config.py`

В файле `nbgrader_config.py` написать

```
c.CourseDirectory.db_students = [
    dict(id="vivanov", first_name="Vovochka", last_name="Ivanov"),
]
```

Проблема По каким-то причинам, `nbgrader` не чувствует изменений этого параметра. Ошибку также не выдает.

5.2.1 Скрипт

С помощью командной строки можно импортировать файл формата `csv` со списком студентов и их контактной информацией (`id`, адрес электронной почты, `lms`).

5.2.2 Командная строка

```
nbgrader db student add vivanov --last-name=Ivanov --first-name=Vovochka #add student's
```

Значение айди может содержать числа и буквы.

Команда `nbgrader db student list` позволяет увидеть текущий список студентов.

Этот метод работает, но файл `nbgrader_config.py` не меняется.

Удалить студента можно с помощью командной строки: `nbgrader db student remove vivanov`.

6 Random useful things

Обычно все неожиданные ошибки связаны с тем, что `nbgrader` ожидает что-то увидеть в корневой папке, и не этого там нет.

Сторона студента(Только еслии у них `nbgrader` стоит) Чтобы студенты ничего не поломали, у них должна стоять 3 и выше версия питона и Jupyter. Чтобы обрабатывать эту ошибку, создатели `nbgrader` советуют добавлять в каждый ноутбук ячейку

```
import IPython
assert IPython.version_info[0] >= 3, "Your version of IPython is too old, please update
```