Load Magic for More Fluid Activities

Sarah M Brown
University of California, Berkeley

Objetives

Learning:

- Reduce typing in short-format workshop to advance to an interesting result in short time
- Focus learners on the content of exercises, reduce syntax burden and context switching

Lesson Development and Workshop Preparation:

- Eliminate > > notation of exercises, simplify management
- Facilitate last minute updates, but still provide learners .zip download

Jupyter Magics

Jupyter Notebooks include a number of built in cell based (%%) and line based (%). magic functions

- %%bash
- %pycat
- %load
- %%writefile

More on Magics: drsmb.cojupmagic
For load magic:

- file extension is implicit
- can use absolute, relative, or url as path
- loads a file for editing and comments the magic

Load Magic Detail

Fig. 2: Before & after load magic

Magic in a Workshop

1. Prepare Learner Repo like 3
2. Distribute .zip download from github (shortened)
3. Teach as usual
4. Introduce load magic as notebook feature
5. Students get exercise as editable text in notebook

Git Workflow for Materials Repo

Git Branches for various stages:

- master: all content
- workshop: hide instructor content for learner download
- day2: add instructor files from part 1, hide all previous
- postworkshop: add all instructor files postworkshop readme

Example workflow notebook: drsmb.co/loadmagicgit

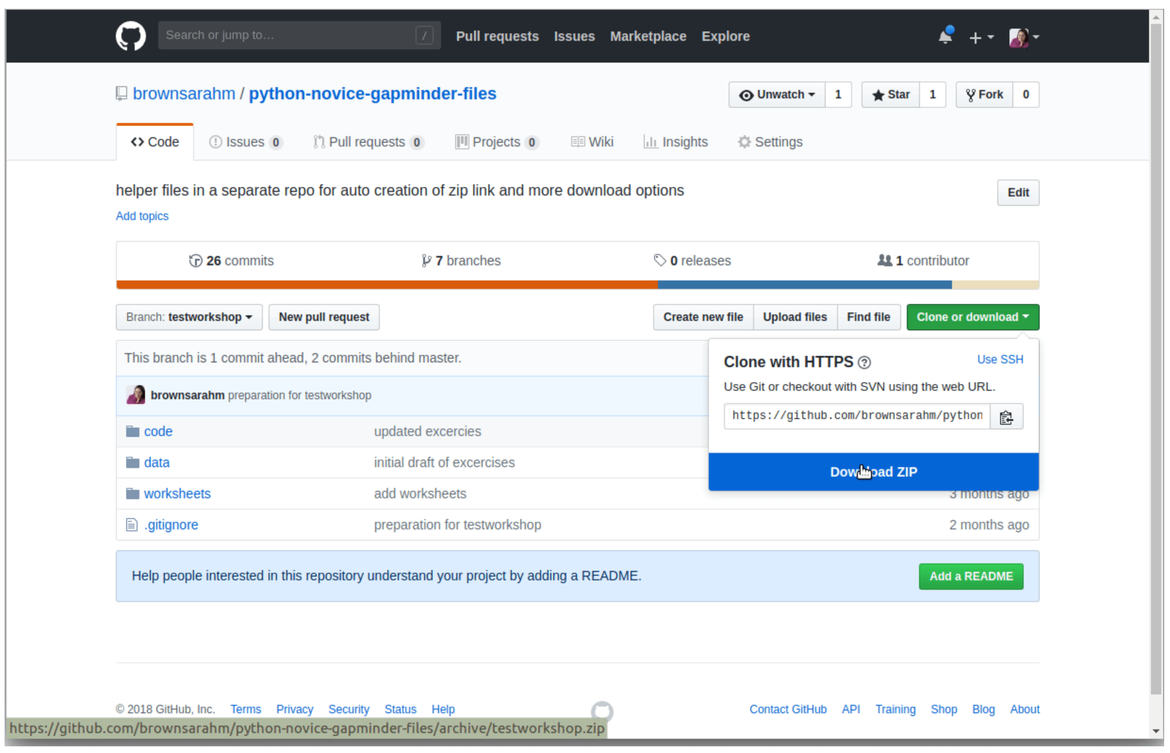
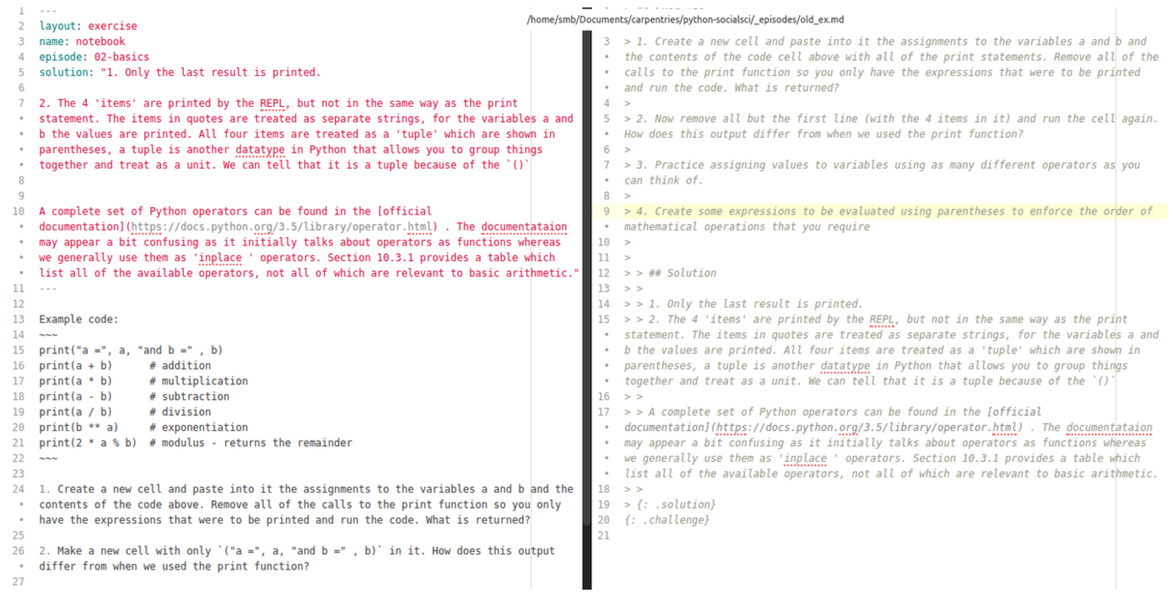


Fig. 3: Github allows providing a link directly to the .zip of a repo. Shorten this (or provide via etherpad) to give students downloads without needing to use git.

Changes to Lesson Organization

To facilitate this teaching method, we can modify how exercises are included in lessons.
Example lesson, partially converted: drsmb.co/magiclesson

1. Add collection to _config.yaml
2. Move exercise content to a Jekyll collection (eg /_exercises/) with one .md file per exercise
3. Add solution as a YAML field of markdown file
4. Incorporate exercises rendered HTML with liquid
5. Add a script to /bin/ that creates <lesson-name>-files repo
6. Instructors customize -files




```
---
layout: exercise
name: notebook
episode: 02-basics
solution: "1. Only the last result is printed.

2. The 4 'items' are printed by the REPL, but not in the same way as the print statement. The items in quotes are treated as separate strings, for the variables a and b the values are printed. All four items are treated as a 'tuple' which are shown in parentheses, a tuple is another datatype in Python that allows you to group things together and treat as a unit. We can tell that it is a tuple because of the `()`


A complete set of Python operators can be found in the [official documentation](https://docs.python.org/3.5/library/operator.html) . The documentataion may appear a bit confusing as it initially talks about operators as functions whereas we generally use them as 'inplace ' operators. Section 10.3.1 provides a table which list all of the available operators, not all of which are relevant to basic arithmetic."
---

Example code:
~~~
print("a =", a, "and b =" , b)
print(a + b)      # addition
print(a * b)      # multiplication
print(a - b)      # subtraction
print(a / b)      # division
print(b ** a)     # exponentiation
print(2 * a % b)  # modulus - returns the remainder
~~~

1. Create a new cell and paste into it the assignments to the variables a and b and the contents of the code above. Remove all of the calls to the print function so you only have the expressions that were to be printed and run the code. What is returned?

2. Make a new cell with only `("a =", a, "and b =" , b)` in it. How does this output differ from when we used the print function?
```

```
> 1. Create a new cell and paste into it the assignments to the variables a and b and the contents of the code cell above with all of the print statements. Remove all of the calls to the print function so you only have the expressions that were to be printed and run the code. What is returned?
>
> 2. Now remove all but the first line (with the 4 items in it) and run the cell again. How does this output differ from when we used the print function?
>
> 3. Practice assigning values to variables using as many different operators as you can think of.
>
> 4. Create some expressions to be evaluated using parentheses to enforce the order of mathematical operations that you require
>
>
> ## Solution
>
> > 1. Only the last result is printed.
> > 2. The 4 'items' are printed by the REPL, but not in the same way as the print statement. The items in quotes are treated as separate strings, for the variables a and b the values are printed. All four items are treated as a 'tuple' which are shown in parentheses, a tuple is another datatype in Python that allows you to group things together and treat as a unit. We can tell that it is a tuple because of the `()`
> >
> > A complete set of Python operators can be found in the [official documentation](https://docs.python.org/3.5/library/operator.html) . The documentataion may appear a bit confusing as it initially talks about operators as functions whereas we generally use them as 'inplace ' operators. Section 10.3.1 provides a table which list all of the available operators, not all of which are relevant to basic arithmetic.
> >
> > {: .solution}
> {: .challenge}
```

Fig. 4: Example of an exercise in proposed and existing formats

```
{% assign cur_ex = site.exercises | where:"name", "notebook" | first %}
{{cur_ex.output}}
```

Fig. 5: Code to include exercise from collection for HTML rendering

```
<blockquote class="challenge">
  <h2 id="exercise">Exercise</h2>

  {{page.content}}
  <blockquote class="solution">
    <h2 id="solution">Solution</h2>

    {{page.solution | markdownify}}
  </blockquote>
</blockquote>
```

Fig. 6: Formatting for Exercises, in _layouts/exercise.html

Lesson Development Benefits to Reorganization

This reorganization will not change the HTML rendering of lessons (See drsmb.co/magicrendered), but it can simplify and create opportunity in lesson development.

- No more double >
- Free staging place for tentative exercises in lesson repos, without rendering in html
- Adds opportunity for more info via YAML (eg: hints, instructor tips, extra challenges)
- Enables exercise only HTML rendering per exercise, episode, or lesson

Points for Discussion

1. Would these style changes help for R lessons?
2. What widgets and features would be helpful in instructor tools?
3. Would extra YAML on exercises improve teaching? eg: hints?

Next Steps

1. Incorporate feedback from discussion
2. Complete more parser and instructor tools
3. Apply changes as necessary to styles repo and submit as PR
4. If accepted, then community can change lessons over gradually or quickly

hello.sarahmbrown.org · smb@sarahmbrown.org