Key binding to interactively execute commands from Python interpreter history in order?

Asked 12 years, 3 months ago Modified 12 years, 2 months ago

Viewed 3k times

2

I sometimes test Python modules as I develop them by running a Python interactive prompt in a terminal, importing my new module and testing out the functionality. Of course, since my code is in development there are bugs, and frequent restarts of the interpreter are required. This isn't too painful when I've only executed a couple of interpreter lines before restarting: my key sequence when the interpreter restart looks like `Up Up Enter Up Up Enter`... but extrapolate it to 5 or more statements to be repeated and it gets seriously painful!

Of course I could put my test code into a script which I execute with `python -i`, but this is such a scratch activity that it doesn't seem quite "above threshold" for opening a text editor :) What I'm really pining for is the `Ctrl-r` behaviour from the bash shell: executing a sequence of 10 commands in sequence in bash involves finding the command in history (repeated `Up` or `Ctrl-r` for a search -- both work in the Python interpreter shell) and then just pressing `Ctrl-o` ten times. One of my favourite bash shell features.

The problem is that while lots of other readline binding functionality like `Ctrl-a`, `Ctrl-e`, `Ctrl-r`, and `Ctrl-s` work in the Python interpreter, `Ctrl-o` does not. I've not been able to find any references to this online, although *perhaps* the `readline` module can be used to add this functionality to the `python` prompt. Any suggestions?

Edit: Yes, I know that using the interactive interpreter is not a development methodology that scales beyond a few lines! But it is convenient for small tests, and IMO the interactiveness can help to work out whether a developing API is natural and convenient, or too heavy. So please confine the answers to the technical question of whether readline history-stepping can be made to work in python, rather than the side-opinion of whether one should or shouldn't choose to (sometimes) work this way!

Edit: Since posting I realised that I am already using the `readline` module to make some Python interpreter history functions work. But the Ctrl-o binding to the `operate-and-get-next` readline command doesn't seem to be supported, even if I put `readline.parse_and_bind("Control-o: operate-and-get-next")` in my `PYTHONSTARTUP` file.

keyboard-shortcuts **python** **readline**

Share

Improve this question

Follow

asked Sep 21, 2012 at 16:34

andybuckley

1,144 ● 3 ● 12 ● 24

Possible duplicate of
stackoverflow.com/questions/12334316/… – Gilles Quénot
Oct 7, 2012 at 22:02

4 Answers

Sorted by: Highest score (default)

4

> I often test Python modules as I develop them by running a Python interactive prompt in a terminal, importing my new module and testing out the functionality.

Stop using this pattern and start writing your test code in a file and your life will be much easier.

- No matter what, running that file will be less trouble.
- If you make the checks automatic rather than reading the results, it will be quicker and less error-prone to check your code.
- You can save that file when you're done and run it whenever you change your code or environment.
- You can perform metrics on the tests, like making sure you don't have parts of your code you didn't test.

Are you familiar with the unittest module?

Share Improve this answer answered Oct 5, 2012 at 12:14

Follow

Mike Graham
76.5k • 16 • 104 • 130

> Yes, I'm aware that putting any substantial tests into a file is much better. But this case was very exploratory -- I was doing slightly different things each time and iterating the API: I just wanted a quick way to re-run the first few "setup" commands each time. Interactive use can help to identify whether the API is natural, so it does have a (limited) place in development. I could have written a setup script and used `python -i`, but that's a bit awkward. So while I appreciate the sentiment, this doesn't answer the particular question of whether history stepping can be done in python! ;)
> – andybuckley Oct 7, 2012 at 17:53

1

Answering my own question, after some discussion on the python-ideas list: despite contradictory information in some readline documentation it seems that the `operate-and-get-next` function is in fact defined as a bash extension to readline, not by core readline.

So that's why `Ctrl-o` neither behaves as hoped by default when importing the `readline` module in a Python interpreter session, nor when attempting to manually force this binding: the function doesn't exist in the readline library to be bound.

A Google search reveals https://bugs.launchpad.net/ipython/+bug/382638, on which the GNU readline maintainer gives reasons for *not* adding this functionality to core readline and says that it should be implemented by the calling application. He also

says "its implementation is not complicated", although it's not obvious to me how (or whether it's even possible) to do this as a pure Python extension to the `readline` module behaviour.

So no, this is not possible at the moment, unless the `operate-and-get-next` function from bash is explicitly implemented in the Python `readline` module or in the interpreter itself.

Share Improve this answer

Follow

answered Oct 8, 2012 at 13:15

andybuckley

1,144 ● 3 ● 12 ● 24

0

This isn't exactly an answer to your question, but if that is your development style you might want to look at DreamPie. It is a GUI wrapper for the Python terminal that provides various handy shortcuts. One of these is the ability to drag-select across the interpreter display and copy only the code (not the output). You can then paste this code in and run it again. I find this handy for the type of workflow you describe.

Share Improve this answer

Follow

answered Oct 7, 2012 at 21:42

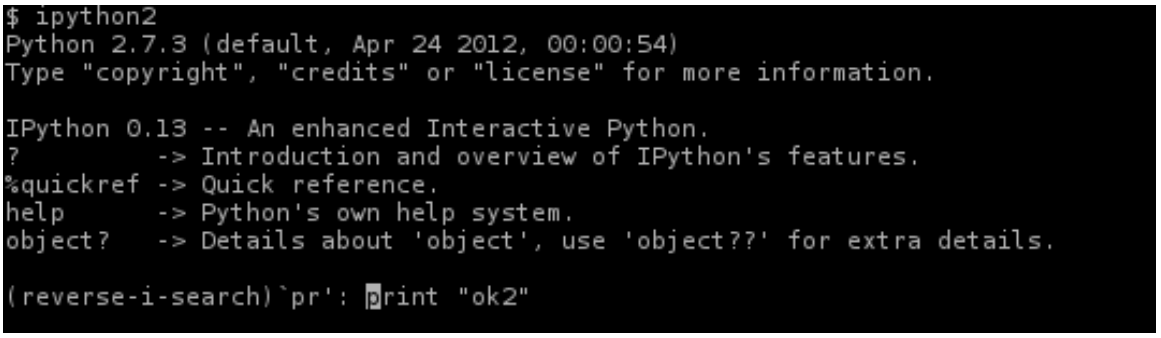
BrenBarn

251k ● 39 ● 418 ● 389

Your best bet will be to check that project : http://ipython.org

0

This is an example with a history search with Ctrl + R :



EDIT If you are running `debian` or derivated :

```
sudo apt-get install ipython
```

Share Improve this answer

Follow

edited Oct 10, 2012 at 1:04

answered Oct 7, 2012 at 22:07

Gilles Quénot

184k ● 43 ● 230 ● 229

The normal python interpreter already supports the same history search. – Johannes Riecken Apr 5, 2019 at 16:10