Getting help with Python

Programming for Biologists

January 13, 2015

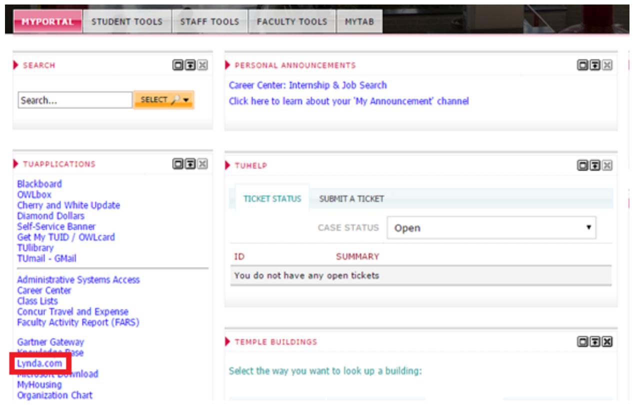
New to Python?

This course will blast through the basics pretty quickly to move on to more biological applications of Python in a timely fashion. This can be overwhelming at first if you're new to the language or computer programming in general. To help you keep up, this guide has collected a few tips and tricks, references, and tutorials. Programming discussion boards are a haven for anyone running into unintelligible errors, and online learning has exploded so much in the last few years that teaching yourself a new coding language is a snap!

Online courses and tutorials

There are lots of online tutorials that allow you to take your time mastering the basics if you want to get a head start before the course or spend extra time outside of class practicing. One of the best is the free Python course on Codecademy. The course offers instantaneous interactive feedback on your code and allows you to progress at your own pace. You can also view discussion boards for each section that other learners contribute their advice and struggles to.

Additionally, as a Temple student you get free access to the online learning database Lynda.com. Make sure to navigate to the page through TUPortal to bypass the paywall.



CASE STATUS: Open

You do not have any open tickets

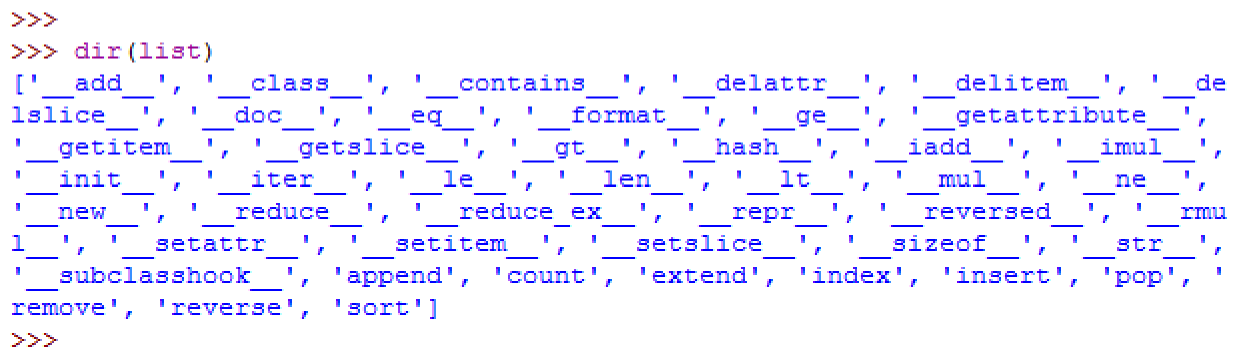
Select the way you want to look up a building:

In particular, there is a course called Up and Running with Python that's designed for beginners using Python 2. In just a few hours, it covers all of the basic syntax of Python that Codecademy goes through, and then continues to more complex programming like working with files or online data. One drawback is that Lynda course appear to be solely based on video without the interactivity of Codecademy exercises.

LearnPython.org is a website that offers a free interactive tutorial with an online input-output coding interface similar to Codecademy. Another option is Python's own tutorial to introduce you to the language. And finally, Dive into Python is a book available freely online. While it's more static than the other tutorials, it is a more comprehensive resource. It may be good to bookmark for future use when you have more in-depth questions or find that another tutorial just hasn't explained a concept well.

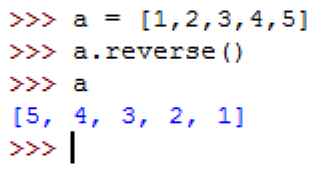
Getting help within IDLE

Two of the first commands you should learn when you start out with Python are `dir()` and `help()`. These magic words will give you more information about objects, commands, modules, and examples of code for commands. `dir()` is a good starting point when you're trying to assess what commands you can apply to a certain object. For example, typing `dir(list)` in the shell will print all the attributes of an object of the class `list` followed by all of the commands that can be applied to a `list`. Not sure if the object you're dealing with is a list? Then use the class function on your object to find out! Simply type `class(a)` in the shell (`a` here is the name of your object).



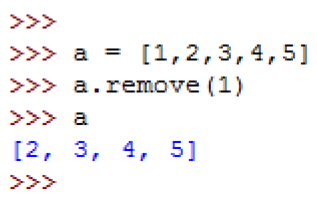
```
>>>
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__de
lslice__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__',
'__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmu
l__', '__setattr__', '__setitem__', '__setslice__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'count', 'extend', 'index', 'insert', 'pop', '
remove', 'reverse', 'sort']
>>>
```

The items that start with double underscores (__) are the attributes of the class, and all of the other words are commands. If you created a list like the one seen in the following example, you could then reverse the order of the items in the list using the `reverse` command that was just revealed by `dir(list)`.



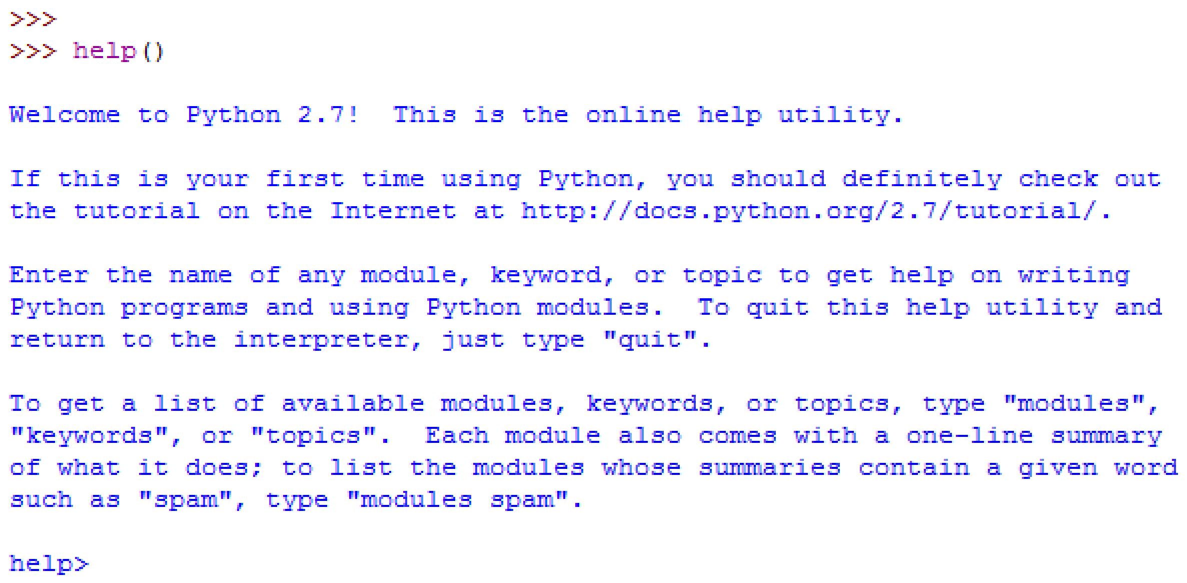
```
>>> a = [1,2,3,4,5]
>>> a.reverse()
>>> a
[5, 4, 3, 2, 1]
>>>
```

The commands listed by `dir()` are commonly executed by typing the name of your object first (”a” in this case was what we assigned to the list) followed by a period and then the command. After the command, there may be arguments to tell the command exactly which parts of the list to act on. In this case there was nothing to specify since there’s only one way to reverse a list and Python already knew which list to work on. For other commands like `remove`, you need to tell Python which list element to remove.



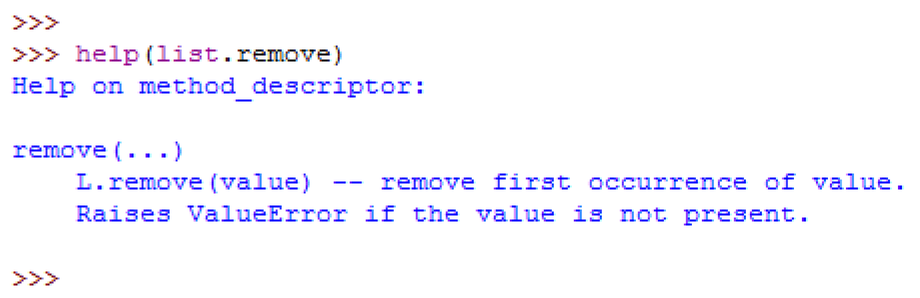
```
>>>
>>> a = [1,2,3,4,5]
>>> a.remove(1)
>>> a
[2, 3, 4, 5]
>>>
```

If you’re unsure of whether a command requires arguments or which arguments mean what, it’s time for the `help()` function. There are two ways to use the `help()` function. The first option is to type just that to start up the searchable help utility.



```
>>>
>>> help()

Welcome to Python 2.7!  This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/2.7/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules",
"keywords", or "topics".  Each module also comes with a one-line summary
of what it does; to list the modules whose summaries contain a given word
such as "spam", type "modules spam".

help>
```

From here you can type keywords into the prompt! Type ”`quit`” or hit enter to exit the help utility. If you are looking for help on one specifice module, command, or class, then you can use the second option. For instance, here is what you would type if you wanted to learn more about the `remove` command for lists:



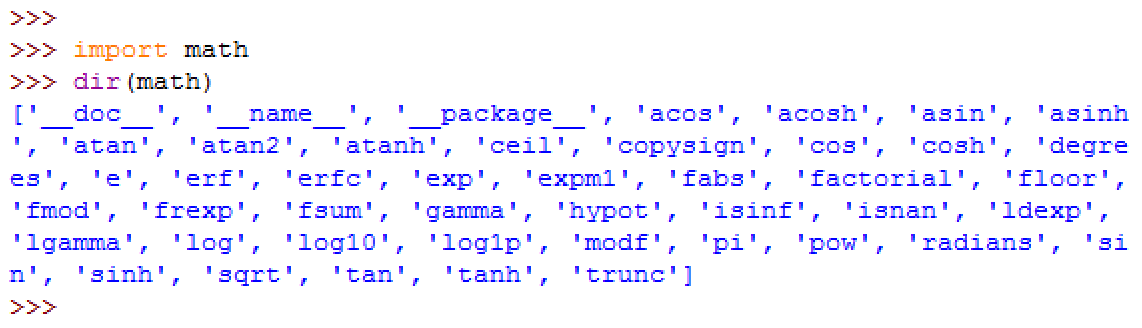
```
>>>
>>> help(list.remove)
Help on method_descriptor:

remove(...)
    L.remove(value) -- remove first occurrence of value.
    Raises ValueError if the value is not present.

>>>
```

The `help()` command gives more detail than `dir()`. The `help` entry will always include the basic definition of an object or usage of a command. In this case, we know for sure now that `remove` can take one or more arguments that are values. The entry also specifies that `remove` will only delete the first occurence of a given value, a very important detail! Idiosyncrasies like these make it a good idea to double check the `help` entry of a command if your code appears to be malfunctioning. Depending on the function or object you search for with `help`, the entry could include more argument definitions, attributes of a class, or samples of code demonstrating how to use the command (a common feature in many modules).

In general, `dir()` is a good starting place to find commands or refresh your memory of all the commands available, and then you proceed to `help()` when you have a specific query you want to know more about.

Getting module-specific help

After you have the basics of Python down, you may still get confused when mastering a new module that brings with it a new set of commands, objects, and rules. `dir()` is once again a helpful first step in exploring this new territory! Typing `dir(`module name`)` will bring up a list of all commands within the general module environment. For example, here's the list produced to show all the functions in the math module that can be written as `math.`function`()`



```
>>>
>>> import math
>>> dir(math)
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh
', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degre
es', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isinf', 'isnan', 'ldexp',
'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'si
n', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
>>>
```

Depending on the results, this can give you likely targets to investigate further with the `help()` function. The more complex the module, the more likely it is that the `help` entries will include long descriptions and code examples.

One of the best sources to get started with a new module is the module's own support page of course. Python is an open source program that allows anyone to create and distribute packages, so the quality of support will vary from module to module. Fortunately the creators of the most popular scientific packages are diligent about this and construct tutorials to guide new users through their code. Follow the links to expore the BioPython, Pyplot (a collection of plotting commands within matplotlib), and NumPy tutorials. Even if the website where you downloaded the module doesn't include a formal tutorial, almost every site has at least some FAQs, examples, or a person to contact should you need assistance. If you are using a Python module that you didn't have to download from a website (like `math` or `csv`) that came included with the installation, Python developers have made sure to create a reference page for that module that includes examples. The complete list is online with links to each package's support page.

When in doubt, ask Google!

Google is a great resource when you start learning a programming language. It instantly connects you to a wealth of knowledge and doesn't judge you for what you don't know. Try to be as specific as possible in your searches, like copying and pasting an error message verbatim that keeps popping up. It's good form to include the word "python" in your search queries to ensure language-specific results. Often several of the first results will be discussion pages on the programming community website Stack Overflow. The contributers here collectively have a huge amount of experience and answer questions from the very basic to the very very obscure. It's almost guaranteed that someone else has run into the same problem and tapped the brilliant minds of the internet to solve it.