Next Steps in Python: Lunch Lessons

with Colby Witherup Wood

and Dan Turner

Lesson 3:
venv, conda, pip, and PEP and chicken
will begin at 12:03 CDT

Please unmute me
(you should hear music)

and special guest Matthew Rich

Next Steps in Python: venv, conda, pip, and PEP

- No Jupyter Notebook today. To access the slides and recipe:
  1. http://www.github.com/agithasnoname/pip_pep
     Click on the green "Clone or download" button, and then Download Zip.

Roast Chicken recipe included in the GitHub repo.

If you have questions at any point during the session, post them in the Zoom chat. Dan will be monitoring.

Lunch Lessons are brought to you by

NUIT Research Computing Services

Have a programming or data question about your research?
We're here to help.
bit.ly/rcsconsult

PEP

Python Enhancement Proposals

https://www.python.org/dev/peps/

PEP

- Some PEPs describe or propose new Python features.
- Informational PEPs provide general guidance to the Python community. Users "are free to ignore Informational PEPs or follow their advice". Some Informational PEPs set guidelines for what is considered "pythonic" and create universal standards for professional Python coders (though each workplace may vary). **It's good to get in the habit of following these standards now.**

PEP 20: The Zen of Python

```
import this
```

Long time Pythoneer Tim Peters summarizes Python's guiding principles into 20 aphorisms, only 19 of which have been written down.

Aphorism - a pithy observation that contains a general truth

PEP 20: The Zen of Python

Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Flat is better than nested. Sparse is better than dense. Readability counts. Special cases aren't special enough to break the rules. Although practicality beats purity. Errors should never pass silently. Unless explicitly silenced. In the face of ambiguity, refuse the temptation to guess. There should be one-- and preferably only one --obvious way to do it. Although that way may not be obvious at first unless you're Dutch. Now is better than never. Although never is often better than *right* now. If the implementation is hard to explain, it's a bad idea. If the implementation is easy to explain, it may be a good idea. Namespaces are one honking great idea -- let's do more of those!

PEP 8: Style Guide for Python Code

- indentation
- where to break up long lines of code
- where to put blank lines in scripts
- how to best import modules
- where to put spaces
- how to name variables

Naming variables and functions

Use lowercase words separated by _

```
animal_list      animals      name_address_dict
sort_animal_list()
```

Naming variables and functions

For temporary variables you can use single letters

```
i       B
```

Avoid using l (lowercase L), O (uppercase o), and I (uppercase i) as single letter variables

Naming files and directories

It is also good to stick to one system for naming files and folders. Camel case is recommended as it can be read on most computer systems and by most programs without problems.

```
animalFile.txt      myScript.py
```

If you use spaces, hyphens, colons, equal signs, slashes, or other special characters, or if your names are too long, you will regret it.

Naming files and directories

If you use spaces, hyphens, colons, equal signs, slashes, or other special characters, or if your names are too long, you will regret it because of

1. differences between Macs and PCs
2. specialized software programs

This includes files that are automatically output by software programs and files that are given to you by collaborators. It is best to rename files before you do anything with them, so they only ever had one name in all your analyses.

Homework

PEP 20 includes recommendations for where to put spaces in your code.

At the end of this slideshow I have added slides about where to put spaces. Go through these slides and then complete the practice exercises in the space_homework.ipynb notebook. There is also an answer key notebook.

Next Steps in Python

Next week: f-strings and the "main" function

Don't forget to register on eventbrite for each week

Need help with installing packages?

We're here to help.
bit.ly/rcsconsult

Have any feedback or suggestions for other Lunch Lesson topics?

Have an easy lunch recipe that you would like to share?

colby.witherup@northwestern.edu

For more Python resources, check out our blog

space homework

Where you should NOT put spaces:

- Immediately inside parentheses, brackets, and braces

  NO: `my_string.split( "," )`

  YES: `my_list = [1, 4, 5]`

space homework

Where you should NOT put spaces:

- Immediately before a comma, colon, or semicolon

NO: `print(list1 , list2)`

NO: `if number > 10 :`

YES: `with open(my_file, "r") as f:`

space homework

Where you should NOT put spaces:

- Between a function and the parenthesis that starts the argument list

  NO: `print (name, date)`

  YES: `str(2005)`

space homework

Where you should NOT put spaces:

- Between a variable and the square bracket that starts its index

  NO: `my_dict [a_key]`

  YES: `my_list[2]`

space homework

Where you should NOT put spaces:

- Inside empty parentheses, brackets, or braces

  NO: `new_dict = { }`

  YES: `new_list = []`

space homework

Where you should NOT put spaces:

- Inside a range

  NO: `my_string[3 : 9]`

  YES: `new_list[0:10]`

space homework

Where you SHOULD put spaces:

- After a comma in a list or tuple

  YES: `print(typeA, typeB, typeC)`

  NO: `new_list = ["cat","dog","rat"]`

space homework

Where you SHOULD put spaces:

- On either side of an operator, unless there are multiple operations…

  YES: `sum_total = num1 + num2`

  NO: `if difference>c-b:`

space homework

Where you SHOULD put spaces:

- On either side of an operator, unless there are multiple operations, in which case you can group by the order of operations

YES: `two_areas = W1*L1 + W2*L2`

NO: `shout = word + "!" * 3`

space homework

Two ways to access the notebook:

1. Go to http://www.github.com/agithasnoname/pip_pep. Click on the green "Clone or download" button, and then Download Zip. Open Anaconda Navigator and choose either Jupyter Lab or Jupyter Notebook. Navigate to the folder you downloaded and open space_homework.ipynb.

2. Go to colab.research.google.com, select GitHub, search for and select agithasnoname/pip_pep. Open space_homework.ipynb.