Python conventions

COMP 364 - Lecture 28
April 2nd, 2012
Mathieu Perreault

Trees are blossoming near the Eiffel Tower in Paris, as temperatures warm up, March 16, 2012. (Joel Saget/AFP/Getty Images) #

Announcements

- Final exam will be on Lectures 15-27 (includes Python)
  - Written part on April 11th 2012
  - Lab part on April 13th, 2012
- Review on Wednesday with the TA (you can ask as many questions as you want!)
  - I will send him a few mock questions.

What we know in Python

- We know a lot of data types (name a few?)
- We saw how to make Python programs (one file)
- We saw how to call functions on specific objects (e.g. strings)
- Today:
  - Making your own functions
  - Putting code in other files and importing it.

Making your own functions

- Making your own functions is advantageous
  - Makes code easier to read
  - Reduce redundancy
- As soon as you end up writing the same portion of code two or more times, consider making it into a function.
- Can you think of procedures you’ve had to write multiple times in the same program?

Anatomy of a function

- Anatomy of a function

  -
    ```
    def functionName(arg1, arg2,...):
      # code here
      return something
    ```

- Arguments: because your function will be created with repeatable behavior in mind, you need some way to control it’s behaviour so that it doesn’t always produce the same thing

- Example of a function:
  ```
  def read_csv_line(line):
    splitLine = line.strip().split(‘,’)
    return splitLine
  ```

How to call your function

- A function needs to be defined before it is used

```
def read_csv_line(myline):
    splitLine = myline.strip().split(‘,’)
    return splitLine
```

- Then, later in code...

```
for line in open(‘myfile.txt’):
  values = read_csv_line(line)
  id = values[0]
  value = values[1]
```

Spotting what should become a function

- Take common procedures and make them into a function

```
menlabel = []
menvalues = []
for line in open('../files/men.txt'):
  label, value = line.strip().split(',')
  value = int(value)
  menlabel.append(label)
  menvalues.append(value)
menbars = [0.1*v for v in menvalues]

womenlabel = []
womenvalues = []
for line in open('../files/women.txt'):
  label, value = line.strip().split(',')
  value = int(value)
  womenlabel.append(label)
  womenvalues.append(value)
womenbars = [0.1*v for v in womenvalues]

plt.figure()
ind = np.arange(len(menlabel))

plt.title('Score by group and gender')
plt.ylabel('Scores')
plt.bar(ind, menvalues, color='r', width=width,yerr=menbars)
plt.bar(ind, womenvalues, bottom=menvalues, color='y', width=width,yerr=womenbars)
plt.xticks(ind+width/2., menlabel)
```

Special way to enter the “main” method

- By default, code that is not in a function will get run (same behaviour that we had before with no functions).
- To make it cleaner, you can add this line:

```
if __name__ == “__main__”:
  # your main code here
```

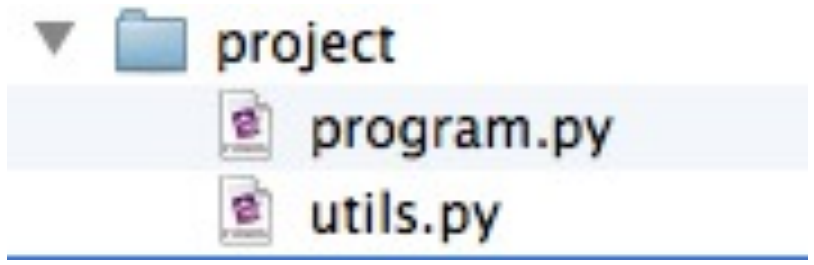
Segmenting code in multiple files

- When you write a lot of code, it may end up in one giant file
- The purpose of functions is to segment better, and you can put those functions in a separate file
  - For example, you created many functions that do plotting
    ```
    make_bar_plot(mylist)
    make_histogram(myvalues)
    ...
    ```
  - You can put them in a file called `plots.py` and import them from any script!
    - `from plots import *`

Your own modules and packages in Python

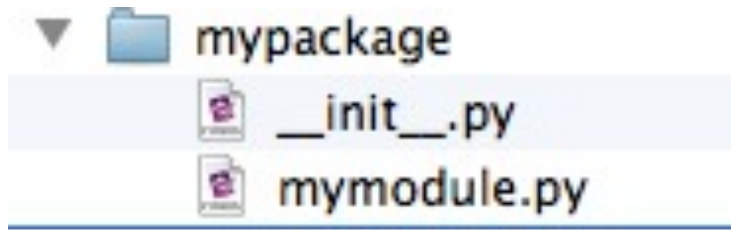
- You can build your own modules!
  - A module is a simple file which you can import if its location is on your Python path.

```
import utils
from utils import myfunction
```



- Build your own packages!
  - Put you code in a folder (e.g. “mypackage”) and create the empty file __init__.py
  - If the “mypackage” folder is in your Python path, you will be able to import it!

```
import mypackage
from mypackage.mymodule import *
```



And in the end...

- *To introduce common computer tools to Life Sciences students in order to help them make sense of their data. Topics include visualization, storage, filtering and analysis.*
- We saw command line, regular expressions, Python, plotting, SQLite
  - Solid toolset for a scientist, put those on your resume, I'll vouch for you!

Thank you!

-m