Code Organization

Python organizes code using *modules* and *packages*.

Generally, modules correspond to files and packages correspond to folders.

You access code and data from different modules and packages by using `import` statements.

http://docs.python.org/tutorial/modules.html

Modules

A *module* is any python source file on your python library path. A file named russia.py will correspond to a module named `russia`, and you would write `import russia` to import the `russia` module into your current scope.

Everything declared in the top level of russia.py would be accessible as an attribute of the module like `russia.COASTLINE_LENGTH`

Packages

A *package* is used for the organization of modules. Any folder on your python path that contains a file name __init__.py is a package, and can contain modules or more packages (sub-folders).

Packages and their sub-packages are dot-separated in Python code, so if we moved russia.py into a folder named "country", our code would then become:

```
import country.russia
import country.liechtenstein

assert country.russia.SIZE > country.liechtenstein.SIZE
```

Partial Import

When you have a number of nested sub packages, it's often inconvenient to use names like `country.russia.industry.EXPORTS`. Instead you can import certain parts of Russia's industry directly into your current scope:

```
from country.russia.industry import EXPORTS
assert 'spices' in EXPORTS
```

```
from country.russia import SIZE as size_of_russia
```

Relative Import

Finally, if you have many nested packages, it's also nice to not have to use the full names in your import statement. For example, we could place the following in country/russia/__init__.py:

```
from .industry import EXPORTS
```