Optimization Redux
x = '12345'
z = '67890'

for a in itertools.product(x, y):
    print(' '.join(a))

Which of the following is not printed?

- '1 6'
- '4 6'
- '6 7'
- '5 0'
x = '12345'
z = '67890'

for a in itertools.product(x, y):
    print(' '.join(a))

Which of the following is not printed?

- '1 6'
- '4 6'
- '6 7' ⭐
- '5 0'
Modules
Many times you need to use a module (package) which has not yet been installed on the machine you are using:

```python
>>> import seaborn

ModuleNotFoundError: No module named 'seaborn'
```
Typically, new packages can be installed using `pip`. 
`pip` is invoked *on the command line*, not within a Python session.

```
$ pip install seaborn
```
Sometimes packages need to be updated because new features are available:

$ pip install --upgrade jupyter
Roll Your Own Module
Creating a Module

Our example: compound interest

\[ A = A_0 \left(1 + \frac{p}{360 \cdot 100}\right)^n \]
Desired usage:

```python
import interest

A0 = 1
p = 5
n = 730
A = interest.present_amount(A0,p,n)
print('$%.2f has compounded to $%.2f after 2 years' % (A0, A))
```
def present_amount( A0, p, n ):
    '''
    Calculate money after compounding A0 money for n days at p percent interest.
    '''
    return A0 * ( 1 + p / ( 360.0 * 100 ) ) ** n
def test_present_amount():
    A0, p, n = 2.0, 5, 730
    A_expected = 2.213398
    A_computed = present_amount( A0, p, n )
    import numpy as np
    assert np.isclose( A_expected, A_computed )
Testing a Module

- At the command line:

  $ pytest interest.py

- Install if needed.
Using a Module

- All code runs at `import`.
- "Hide" script-like code using a construct like this:

```python
if __name__ == '__main__':
    test_present_amount()
    test_initial_amount()
    test_days()
    test_annual_rate()

    print( present_amount( 100,10,365 ) )
```
More Python
Comparing results

- Pandas—Python for Data Analysis
- Scikit-Learn—machine learning
- Classes (object-oriented programming)
- SymPy—symbolic algebra
- Bokeh—interactive plots, like web graphics