University of Wisconsin
La Crosse
Computer Science

CS 224 Introduction to Python

Packages

1

## Consider these modules

Cars

```
engine.py
transmission.py
electrical.py
brakes.py
unibody.py
```

Trucks

```
engine.py
transmission.py
electrical.py
brakes.py
frame-body.py
```

Motorcycles

```
engine.py
transmission.py
electrical.py
brakes.py
frame.py
```

We need to use all of these in a single project but the names collide.

2

## Possible Solutions

- Use `import as` to rename the modules at runtime
  - This is rather ad hoc
- Rename the modules by changing filenames
  - Not ideal: filenames were chosen to be descriptive
- Find work in another industry
  - It won't pay as well as CS (unless you go into business but the thought of doing that makes your blood run cold)
- Manage the modules in a package
  - Please, Dr. Mathias, tell us more!

3

## What is a package?

- It's just a collection of modules
  - (see slides from Class 35)
- It can be hierarchical
  - i.e. contain subpackages
  - a subpackage is a package within another package
- Can contain (limited) runnable code outside of the modules
- It's implemented as a directory on disk
  - package name is determined by directory name
  - (analogous to a module name being determined by .py filename)

4

# Let's package-ize our example

```
Vehicles/
    Cars/
        engine.py
        transmission.py
        electrical.py
        brakes.py
        unibody.py
    Trucks/
        engine.py
        transmission.py
        electrical.py
        brakes.py
        frame-body.py
    Motorcycles/
        engine.py
        transmission.py
        electrical.py
        brakes.py
        frame.py
```

Top-level directory and package name

Subdirectories and subpackages

If you're a Windows user:
1. I'm sorry
2. Replace '/' with '\'

5

# Can nest more deeply as needed

```
Vehicles/
    Cars/
        Drivetrain/
            engine.py
            transmission.py

        electrical.py
        brakes.py
        unibody.py
```

Subpackage of subpackage Cars

Similar to functional decomposition, logical structure of your problem will determine the hierarchy of your packages

6

## One more detail

A package can (should?) include an __init__.py file to execute initialization code:

```
Vehicles/
    __init__.py
    Cars/
        __init__.py
        Drivetrain/
            __init__.py
            engine.py
            transmission.py

        electrical.py
        brakes.py
        unibody.py
```

More on this in a moment

7

## Using a package

- To use the package, `import` some or all of its modules in another program

```
Vehicles/
    Cars/
        Drivetrain/
    Trucks/
    Motorcycles/
```

```
import Vehicles.Cars.brakes
Vehicles.Cars.brakes.absTest()

from Vehicles.Cars import brakes
brakes.absTest()

from Vehicles.Cars.brakes import absTest
absTest()
```

8

## Using a package

- When you import any part of a package, its `__init__.py` is executed

```
Vehicles/
    __init__.py
    Cars/
        __init__.py
        Drivetrain/
            __init__.py
    Trucks/
        __init__.py
    Motorcycles/
        __init__.py
```

```
import Vehicles.Cars.brakes
Vehicles.Cars.brakes.abs_test()

from Vehicles.Cars import brakes
brakes.abs_test()

from Vehicles.Cars.brakes import abs_test
abs_test()
```

All of these cause: Vehicles/__init__.py and Vehicles/Cars/__init__.py to execute

9

## Importing an entire package

```
import Vehicles

Vehicles.Cars.brakes.abs_test()        # ERROR
```

This doesn't import anything unless…

10

# __init__.py

- Because `__init__.py` is run when we load a module, we can use it to control imports

```
# Vehicles/__init__.py

from . import Cars, Trucks, Motorcycles
```

```
# Vehicles/Trucks/__init__.py

from . import engine, transmission, electrical, brakes
```

11

# __init__.py

```
# Vehicles/__init__.py

from . import Cars, Trucks, Motorcycles
```

```
# Vehicles/Trucks/__init__.py

from . import engine, transmission, electrical, brakes
```

```
import Vehicles          # now this call imports the modules
                         # enumerated in the init files
```

12

## Wildcard `import`

- Recall that in a module, we can define `__all__` to control the elements that are imported using *
- Similarly, we can define __all__ in a package to control the modules that are imported using *

```
# Vehicles/Trucks/__init__.py

__all__ = [engine, transmission, electrical, brakes]
```

```
# code that uses the package

from Vehicles.Trucks import *
```

13

## Relative `import`

- Some modules in a package may need to import elements from other modules in that package

```
# Vehicles/Trucks/transmission.py

from ..Vehicles.Cars import unibody
```

.. indicates up one level

14