LECTURE 20

MODULES

MCS 260 Fall 2020

Emily Dumas

REMINDERS

- Project 2 due today at 6:00pm central
- Worksheet 7 available
- Quiz 7 due Monday—multiple work sessions OK

WHAT MODULES DO

A complex Python program usually has many lines.

Keeping them in one file is quite limiting. Very long source files are hard to navigate and understand.

Modules are Python's solution. They let you spread code across multiple files.

MODULES SO FAR

We discussed the `math`, `random`, `sys`, and `os` modules. The pattern is the same: Import the module to make it available, use the functions as *module_name.fn_name*(...).

```
import sys
import os

if os.path.exists("out.dat"):
    print("Error: out.dat already exists; not overwriting.")
    sys.exit()

f = open("out.dat","w")
# ...
```

MAKING YOUR OWN MODULE

```
import foo
```

will look for a module named "foo" in the current directory and several other places. The list of places is stored in `sys.path`.

For example if `foo.py` exists in the current directory, it will be imported by this command.

WHAT IMPORTING MEANS

Functions and variables from the module are made available with the module name as a prefix, e.g. `doit()` becomes `foo.doit()`.

Code in the module outside any function is executed. Usually, files designed to be used as modules have no code other than functions and global variables.

EXAMPLE: DICE GAME

WHY TO USE MODULES

- **Reusability**: The same module can be used by many programs.
- **Isolation**: No conflict between function and var names in module and those in program.
- **Implementation hiding**: Can substitute any module which accomplishes the same tasks (e.g. more efficiently) with no change to main program.

WHAT TO MOVE INTO A MODULE

- Functions with related purpose.
- Functions that call each other, but nothing from the rest of the program.
- Functions whose purpose is significantly more general from the program you are developing.

OTHER IMPORT SYNTAX

If you want to, it is possible to import a few functions from a module into the global namespace, i.e. so the module name need not be used when calling them.

```
import random

print("Random digit: ",random.randint(0,9))
```

OTHER IMPORT SYNTAX

If you want to, it is possible to import a few functions from a module into the global namespace, i.e. so the module name need not be used when calling them.

```
from random import randint

print("Random digit: ",randint(0,9))
```

Import single name to global namespace:

```
from module import name
```

Import multiple names to global namespace:

```
from module import name0, name1, name2
```

Import all names to global namespace:

```
from module import *
```

ADVICE

`import foo` is almost always better than `from foo import ...`

RULE

You are not allowed to use `from foo import ...` in code submitted to MCS 260 assignments.

PROGRAMS THAT WORK AS MODULES

It is convenient to write programs that do something when run on their own, but which only define functions when imported. This makes it easier to test functions in the REPL, for example.

```
# no_main_wrap.py
# Importing this will run the main loop

def f(x):
    """polynomial function"""
    return 2.0*x**3 - 3.0*x**2

# Main loop
for i in range(11):
```

```
    t = i/10
    print("f({}) = {}".format(t,f(t)))
```

```python
# main_wrap.py
# Importing this will not run anything

def f(x):
    """polynomial function"""
    return 2.0*x**3 - 3.0*x**2

def main():
    # Main loop
    for i in range(11):
        t = i/10
        print("f({}) = {}".format(t,f(t)))

if __name__=="__main__":
    # We are the main program, not an import
    main()
```

REFERENCES

- In *Downey*:
  - Section 14.9 discusses writing modules

REVISION HISTORY

- 2020-10-08 Initial publication