CISS450: Artificial Intelligence
Lecture 14: Modules

Yihsiang Liow

Agenda

- Study modules, a way of organizing code in Python

Module

- You can think of a module as a .py file that is loading into memory
- You have already seen this. Example:

```
import math # NOT "import math.py"!!!
print(math.pi)
```

- When you issue the `import math` statement,
  - the Python interpreter looks for `math.py` and execute the commands in that file.
  - Identifiers created are places in a namespace called `math`. So `math.pi` refers to the `pi` variable defined in `math.py` but is placed in the `math` namespace

Modules

- In Python, a namespace is just a map between names and objects
- The ***main*** or ***top-level*** module is the module that you start with

Module Search Path

- The Python interpreter looks for modules in specific places only, ***not*** throughout the whole hard drive!
- Try: Create an `x.py` with the following code and save it onto the Desktop:

```
# x.py
x = 0
```

Now at the shell, try:

```
import x
```

Did the Python interpreter find `x.py`?

Module Search Path

- Try this:

```
import sys
print(sys.path)
```

- The Python interpreter search for modules in the current directory, then `sys.path`
- So if you have files in `/home/jdoe/project` and you want the Python interpreter to search there, then you should put that in `sys.path`

```
import sys
sys.path.append("/home/jdoe/project")
# now import your module ...
```

Compiled Python Bytecode

- When a `.py` is imported, a compiled `.py` file is created. This file has extension `.pyc`. The compiled python bytecode file will then be used by future imports to avoid re-compilation.
- WARNING: If you've made changes to a `x.py` file and your shell has already imported `x.py`, importing it again will ***not*** import `x.py`. To "re-import", use the `reload` command:

```
import x # first import
reload(x)
```

Changing Namespace

- You can change the namespace:
- Try:

```
import math as m # lazy???
print(m.pi)
```

- WARNING:

```
import math as m
reload(math) # WRONG!
reload(m)    # right
```

Importing Multiple Modules

- More generally you can import several modules at the same time:
- Example:

```
import math, sys
print(math.pi)
print(sys.path)
```

Importing Into Current Namespace

- You can also put all the names into your current namespace to save typing:

```
from math import *
print(pi)
```

- WARNING: This will overwrite the names already defined (if any)!!!
- To prevent this you can import selectively:

```
from math import pi
print(pi)
```

- More generally:

```
from math import pi, sin
print(sin(pi))
```

Importing Into Current Namespace

- In general, it's a BAD practice to import into the current namespace, especially if not imported selectively. This is called ***namespace pollution*** or ***flattening namespaces***.
- Of course another way to save typing is to define new names in your current namespace:

```
import math
pi = math.pi
```

`__name__`

- Try: Create `test1.py` with the following:

```
print("test.py")
print("__name__:", __name__)
```

  Now create `test2.py` in the same directory as `test1.py` with the following:

```
print("__name__:", "__name__)
print("importing test")
import test
```

- What do you see? Explain!
- Here's how to use `__name__` ...

__name__

- Create `avg.py`:

```
def max(*xs):
    m = xs[0]
    for x in xs[1:]:
        if m < x: m=x
    return m

if __name__=="__main__":
    print("testing max ...")
    if max(3,5,2) == 5: print("ok")
    else: print("error")
```

- Run `avg.py`. From shell (or another `.py` file), import avg. Does the test run?

Resources

- Go to your best friend and look for the Python Tutorial written by Guido van Rossum and Fred Drake. Search for “Modules”. Read it.
- The above article is a very well-written article for programmers. You definitely want to read the whole article when you have time. (Recall: Guido is the author of Python.)