# Ch12-ModulesAndPackages

September 22, 2021

## 1 Modules and Packages

http://openbookproject.net/thinkcs/python/english3e/modules.html - module is a file containing Python definitions and statements intended for use in other Python programs - standard library is an example of Python language provided modules

### 1.1 Various ways to import names into the current namespace

[ ]:
```
# import math module into the global namespace
import math
x = math.sqrt(100)
print(x)
```

[ ]:
```
import random
print(random.choice(list(range(1, 21))))
```

[ ]:
```
from random import choice
```

[ ]:
```
print(choice([1, 2, 3, 4]))
```

[ ]:
```
help(math)
```

[ ]:
```
from math import * # Import all the identifiers from math
print(sqrt(100))
print(pi)
```

[ ]:
```
from math import radians, sin
rad = radians(90)
print(rad)
print(sin(rad))
```

### 1.2 names can be imported in to local namespace

[ ]:
```
def isUpper(letter):
    import string # string name is local
    return letter in string.ascii_uppercase
```

[15]:
```
print(isUpper('a'))
```

```
False
```

[2]:
```
# can we use string module outside isUpper function?
print(string.digits)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-1f51304bf154> in <module>()
      1 # can we use string module outside isUpper function?
----> 2 print(string.digits)

NameError: name 'string' is not defined
```

### 1.3 scope and lookup rules

The scope of an identifier is the region of program code in which the identifier can be accessed, or used.

Three important scopes in Python: - Local scope refers to identifiers declared within a function - Global scope refers to all the identifiers declared within the current module, or file - Built-in scope refers to all the identifiers built into Python – those like range and min that are (almost) always available

Precedence rule:

innermost or local scope

global scope

built-in scope

[1]:
```
def testLocalScope():
    k = 5
    for i in range(2):
        for j in range(2):
            if i == j:
                k = 3
                print('inside=',k)
    print('outside=',k)
    print(i, j, k)
```

[3]:
```
testLocalScope()
```

```
inside= 3
inside= 3
outside= 3
1 1 3
```

[21]:
```
# can't access k outside the testLocalScope function
print(k)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-eb2fa875d160> in <module>()
----> 1 print(k)

NameError: name 'k' is not defined
```

### 1.4 User-defined modules

- see `modules` folder
- see main.py and module2.py inside modules folder
- demonstrates user defined modules and importance of import guard
- run each module, but main.py depends on module2.py

```
if __name__ == '__main__':
    ...
```

## 2 Packages

- folder with module(s)
- must define *__init__*.py empty module to initialize as package
- can't import package itself (in a useful way) but only module(s) or identifiers in the modules
- https://docs.python.org/3/tutorial/modules.html#packages

### 2.1 fibos package

- the folder fibos in this repository is an example of Python package
- take a look inside the packate and observe the files
- see demo script `demos/package_demo.py` that uses fibos package
- the following code snippets demonstrate using user-defined package

[4]:
```
# change current working directory to demos
%cd demos
```

```
/Users/rbasnet/CMU/projects/Python-Fundamentals/demos
```

[5]:
```
import fibos
```

[6]:
```
help(fibos)
```

```
Help on package fibos:

NAME
    fibos
```

```
PACKAGE CONTENTS
    fibo

FILE
    /Users/rbasnet/CMU/projects/Python-Fundamentals/demos/fibos/__init__.py
```

[7]:
```
# can't use the imported package to access its modules!
fibos.fibo.fib(10)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-7-a690191fd105> in <module>
      1 # can't use the imported package to access its modules!
----> 2 fibos.fibo.fib(10)

AttributeError: module 'fibos' has no attribute 'fibo'
```

[8]:
```
# must import the modules or identifiers defined in the package
import fibos.fibo as f
f.fib(10)
```

```
0 1 1 2 3 5 8 13 21 34
```

[9]:
```
from fibos import fibo
fibo.fib2(10)
```

[9]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

[ ]:
```

```