ESCI 386 – Scientific Programming, Analysis and Visualization with Python

Lesson 9 - Modules and Functions

Defining and Using a Function

```
def sphere_area(r):
    pi = 3.1415927
    return 4*pi*r**2
```

```
>>> sphere_area(5)
314.15927
```

Using Keywords

```
def sphere_area(r, radius = True):
    pi = 3.1415927
    if radius:
        pass
    else:
        r = r/2.0
    return 4*pi*r**2
```

```
>>> sphere_area(5)
314.15927
>>> sphere_area(5,radius = False)
78.5398175
```

Lambda Operator

- Python also has a simple way of defining a one-line function.
- These are created using the Lambda operator.
- The code must be a single, valid Python statement.
- Looping, if-then constructs, and other control statements cannot be use in Lambdas.

```
>>> bar = lambda x,y: x + y
>>> bar(2,3)
5
>>> cube_volume = lambda l, w, h: l*w*h
>>> cube_volume(2, 4.5, 7)
63.0
```

Modules

- A module is simply a collection of functions and other Python statements that can be loaded using the `import` command, and then accessed.
- In addition to functions being defined within modules, constants can also be defined.
- The functions and constants are accessed by prefacing them with the name of the module.

sys.path Variable

- When importing a module, Python searches through the directories listed in the sys.path variable, as well as through the current working directory.

- To import modules located in other directories you need to add that directory to the sys.path variable. This is done via the following two lines:

```
import sys
sys.path.append(dirname)
```

- In the above lines, dirname is the full name of the directory that you want added to sys.path.

Module Example

```
pi = 3.1415927

def area(r):
   return 4.0*pi*r**2

def volume(r):
   return (4.0/3.0)*pi*r**3
```

$\longrightarrow$ Saved to ‘Sphere.py’

```
>>> import Sphere
>>> Sphere.pi
3.1415927
>>> Sphere.area(5)
314.15927
>>> Sphere.volume(5)
523.5987833333334
```

Namespace Consideration

- We can import the module using a shorter alias if we want.

```
>>> import Sphere as s
>>> s.area(4)
201.0619328
```

Namespace Consideration (cont.)

- We can choose to import only specific functions or constants from a module.

```
>>> from Sphere import area
>>> area(4)
201.0619328
```

- We can also rename them as we import them.

```
>>> from Sphere import area as a
>>> a(4)
201.0619328
```

Namespace Consideration (cont.)

- We can use * to import all constants and functions from a module without having to use the module name.

```
>>> from Sphere import *
>>> area(4)
201.0619328
```

- However, this is <u>not recommended</u>!

- You might inadvertently import two functions with the same name from two different modules, which can cause confusion.

Documentation Strings

- A document string is an optional string at the beginning of a module or function that describes the module or function and its use.

- The document string can be accessed via the `__doc__` attribute.

- It is a good idea to include at least a one-line document string in your functions and modules.
  - The documents string should show usage and explain any options.

Documentation String Example

```
def sphere_area(r, radius = True):
    '''sphere_area(r, radius = True)
    If radius == True, r is a radius
    Otherwise, r is diameter.'''

    pi = 3.1415927
    if radius:
        pass
    else:
        r = r/2.0
    return 4*pi*r**2
```

```
>>> print(sphere_area.__doc__)
sphere_area(r, radius = True)
    If radius == True, r is a radius
    Otherwise, r is diameter.
```