Programming using Python

Modules and Packages

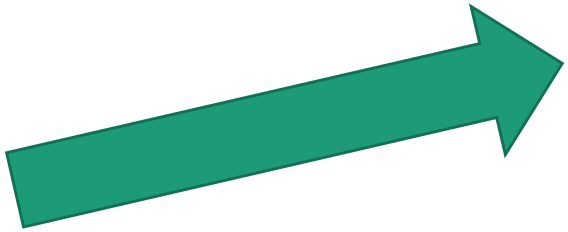
Modules

- As program gets longer, need to organize them for easier access and easier maintenance.
- Reuse same functions across programs without copying its definition into each program.
- Python allows putting definitions in a file
  - use them in a script or in an interactive instance of the interpreter
- Such a file is called a *module*
  - definitions from a module can be *imported* into other modules or into the *main* module

Modules

- A module is a file containing Python definitions and statements.
- The file name is the module name with the suffix .py appended.
- Within a module, the module's name is available in the global variable __name__.

Modules Example



fib.py - C:\

fib.py - C:\Users\karkare\Google Drive\IITK\Courses\2016Python\Programs\fib.py (2.7.12)

File Edit Format Run Options Window Help

```
# Module for fibonacci numbers

def fib_rec(n):
    '''recursive fibonacci'''
    if (n <= 1):
        return n
    else:
        return fib_rec(n-1) + fib_rec(n-2)
```

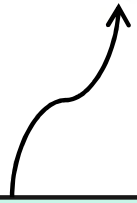
Modules Example

```
def fib_rec(n):
    '''recursive fibonacci'''
    if (n <= 1):
        return n
    else:
        return fib_rec(n-1) + fib_rec(n-2)

def fib_iter(n):
    '''iterative fibonacci'''
    cur, nxt = 0, 1
    for k in range(n):
        cur, nxt = nxt, cur+nxt
    return cur

def fib_upto(n):
    '''given n, return list of fibonacci
    numbers <= n'''
    cur, nxt = 0, 1
    lst = []
    while (cur < n):
        lst.append(cur)
        cur, nxt = nxt, cur+nxt
    return lst
```

```
>>> import fib
>>> fib.fib_upto(5)
[0, 1, 1, 2, 3]
>>> fib.fib_rec(10)
55
>>> fib.fib_iter(20)
6765
>>> fib.__name__
'fib'
```



Within a module, the module's name is available as the value of the global variable __name__.

Importing Specific Functions

- To import specific functions from a module

- This brings only the imported functions in the current symbol table
  - No need of modulename. (absence of `fib.` in the example)

Importing ALL Functions

- To import *all* functions from a module, in the current symbol table

```
>>> from fib import *
>>> fib_upto(6)
[0, 1, 1, 2, 3, 5]
>>> fib_iter(8)
21
```

- This imports all names except those beginning with an underscore (_).

`__main__` in Modules

- When you run a module on the command line with

  ```
  python fib.py <arguments>
  ```

  the code in the module will be executed, just as if you imported it, but with the `__name__` set to "__main__".

- By adding this code at the end of your module

  ```
  if __name__ == "__main__":
      ... # Some code here
  ```

  you can make the file usable as a script as well as an importable module

`__main__` in Modules

```
if __name__ == "__main__":
    import sys
    print (fib_iter(int(sys.argv[1])))
```

- This code parses the command line only if the module is executed as the “main” file:

```
$ python fib.py 10
55
```

- If the module is imported, the code is not run:

```
>>> import fib
>>>
```

Package

- A Python package is a collection of Python modules.
- Another level of *organization.*
- *Packages* are a way of structuring Python's module namespace by using *dotted module names*.
  - The module name A.B designates a submodule named B in a package named A.
  - The use of dotted module names saves the authors of multi-module packages like NumPy or Pillow from having to worry about each other's module names.

A sound Package

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

https://docs.python.org/3/tutorial/modules.html

A sound Package

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

What are these files with funny names?

https://docs.python.org/3/tutorial/modules.html

`__init.py__`

- The `__init__.py` files are required to make Python treat directories containing the file as packages.
- This prevents directories with a common name, such as `string`, unintentionally hiding valid modules that occur later on the module search path.
- `__init__.py` can just be an empty file
- It can also execute initialization code for the package

Importing Modules from Packages

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

https://docs.python.org/3/tutorial/modules.ht

Importing Modules from Packages

`import sound.effects.echo`

- Loads the submodule `sound.effects.echo`
- It must be referenced with its full name:

```
sound.effects.echo.echofilter(
    input, output,
    delay=0.7, atten=4
)
```

Importing Modules from Packages

`from sound.effects import echo`

- This also loads the submodule `echo`
- Makes it available without package prefix
- It can be used as:

```
echo.echofilter(
    input, output,
    delay=0.7, atten=4
)
```

Importing Modules from Packages

```
from sound.effects.echo import echofilter
```

- This loads the submodule `echo`, but this makes its function `echofilter()` directly available.

```
echofilter(input, output,
           delay=0.7, atten=4)
```

Popular Packages

- pandas, numpy, scipy, matplotlib, …
- Provide a lot of useful functions