7.Modules

2017/11/30 18:25

- usage
  - to split long script into several files for easier maintenance.
  - to use a handy function that you've written in several programs without copying its definition into each program.
- def
  - A module is a file containing Python definitions and statements.
  - the module name is the file name, e.g. xxx.py
  - Within a module, the module's name (as a string) is available as the value of the global variable `__name__`.
- how to use
  - `>>> import module_name`
  - `module_name.function  #` to access the functions inside
  - `a = module_name.function  #` If you intend to use a function often you can assign it to a local name
- Content
  - its own private symbol table, which is used as the global symbol table by all functions defined in the module
    - to touch a module's global variables, `modname.itemname`.
  - executable statements
    - These statements are intended to initialize the module. They are executed only the *first*time the module name is encountered in an import statement
  - as well as function definitions
  - other modules
- import
  - `import` statements is better at the beginning of a module (or script)
  - e.g.
    - ```
      import foo                  # foo imported and bound locally
      import foo.bar.baz          # foo.bar.baz imported, foo bound locally
      import foo.bar.baz as fbb   # foo.bar.baz imported and bound as fbb
      ```
      ```

```
from foo.bar import baz     # foo.bar.baz imported and
bound as baz
from foo import attr        # foo imported and foo.attr
bound as attr
```

- special import
  - imports names from a module directly into the importing module's symbol table.
  - e.g. `>>> from fibo import fib, fib2`
    - `fibo` is not defined but fib & fib2 are introduced
  - e.g. `>>> from fibo import *`
    - This imports all names except those beginning with an underscore (_).
    - better not to use this since it introduces an unknown set of names into the interpreter

- The Module Search Path
  - the interpreter first searches for a built-in module with that name.
  - If not found, it then searches for a file named `spam.py` in a list of directories given by the variable `sys.path`.
    - `sys.path` is initialized from these locations:
      - The directory containing the input script
      - `PYTHONPATH` (a list of directory names, with the same syntax as the shell variable `PATH`).
      - The installation-dependent default.

- Standard Modules
  - One particular module deserves some attention: `sys`, which is built into every Python interpreter.
    - `sys.ps1` and `sys.ps2` define the strings used as primary and secondary prompts
    - `sys.path` is a list of strings that determines the

interpreter's search path for modules.

- `>>> sys.path.append('/ufs/guido/lib/python')`

- The `dir()` Function
  - The built-in function `dir()` is used to find out which names a module defines. It returns a sorted list of strings