Modules

April 28, 2020

1 Modules

2 The Evolution of Our Love for Python

- Interactive Mode

- Script Mode
- Functions

- Modules
  - Logically organize Python code by grouping related code into a python file or module.
  - As program gets longer, split it into several files for easier maintenance. or,
  - Use a handy function that you've written in several programs without copying its definition into each program.

3 Modules

- A module is a python file containing **logically related** python code
  - file_name = module_name.**py**
- `dir()` is used to find out what definitions a module include

[3]:
```
import math
print(dir(math))
```

```
[ __doc__ ,  __file__ ,  __loader__ ,  __name__ ,  __package__ ,  __spec__ ,
 acos ,  acosh ,  asin ,  asinh ,  atan ,  atan2 ,  atanh ,  ceil ,  copysign ,
 cos ,  cosh ,  degrees ,  e ,  erf ,  erfc ,  exp ,  expm1 ,  fabs ,
 factorial ,  floor ,  fmod ,  frexp ,  fsum ,  gamma ,  gcd ,  hypot ,  inf ,
 isclose ,  isfinite ,  isinf ,  isnan ,  ldexp ,  lgamma ,  log ,  log10 ,
 log1p ,  log2 ,  modf ,  nan ,  pi ,  pow ,  radians ,  sin ,  sinh ,  sqrt ,
 tan ,  tanh ,  tau ,  trunc ]
```

4 Modules - `imp rt`

- A module need to be **imported** before use. A module can be imported into
  - other modules, or
  - the **main** module, or
  - an interactive instance of the interpreter(shell)
- Variants of `import`:
  - `import` module: import module(not functions), then use module name to access functions with `.` notation as `module.function`
  - `from` module `import` function: import specific functions, then access functions directly
  - `from` module `import *`: import all functions, then access functions directly

5 Modules - Revisit Fibonacci Series

- Two version of fib(n): print vs. return (see definition in Functions)
- Logically related and can be organized in a module `fibo` in file fibo.py

[ ]:
```
#fibo.py
def fib_print(n): # Define function - write Fibonacci series up to n
    """Print a Fibonacci series up to n."""# Documentation string
    a = 0; b = 1; # set up base
    while b < n:  # condition
        print(b, end=   ); c = a + b; a = b; b = c

def fib_return(n): # Define function - write Fibonacci series up to n
    """Retunr a list containing a Fibonacci series up to n."""# Documentation
 ↪string
    series = []#create an emply list for holding the series
    a = 0; b = 1; # set up base
    while b < n:  # condition
        series.append(b); c = a + b; a = b; b = c
    return series
```

[11]:
```
import fibo #import module, not functions
fibo.fib_print(20) # use module to access function as module.function
```

[6]:
```
from fibo import fib_return#import function fib_return ONLY from module
print(fib_return(20))# access function directly
```

```
[1, 1, 2, 3, 5, 8, 13]
```

[8]:
```
from fibo import * #import all functions fib_return from module
fib_print(20) # access any function directly
```

```
print(fib_return(20))
```

```
1 1 2 3 5 8 13 [1, 1, 2, 3, 5, 8, 13]
```

6 Standard Modules

- Python comes with a library of standard modules. For a breif tour- https://docs.python.org/3/tutorial/stdlib.html#mathematics
- Commonly used ones: `math`, `datetime`

7 Modules - `Rel ad`

- `import` statements are executed only the **first** time the module name is encountered
- For any module update after first `import`, use `reload`

[14]:
```
import importlib
importlib.reload(fibo)
```

[14]:
```
<module fibo from /home/nbuser/library/Master/Lecture/fibo.py >
```

8 Standard Modules - `math`

- The `math` module gives access to functions for floating point math
- https://docs.python.org/3/library/math.html

[12]:
```
import math #import math module
print(math.cos(math.pi / 4)) # use math.cos and math.pi
print(math.log(1024,2)) # use math.log
help(math.ceil) #find help on math.ceil
```

```
0.7071067811865476
10.0
Help on built-in function ceil in module math:

ceil(...)
    ceil(x)

    Return the ceiling of x as an Integral.
    This is the smallest integer >= x.
```

9 Standard Modules - `datetime`

- The `datetime` module supplies classes for manipulating dates and times in both simple and complex ways.
- https://docs.python.org/3/library/time.html

[13]:
```
import datetime
now = datetime.date.today()
print(now)
birthday = datetime.date(1994, 7, 31)# dates support calendar arithmetic
age = now - birthday
age.days
```

```
2020-04-28
```

[13]:
```
9403
```

[ ]:
```

```