# 6.7 程序结构

- 书的内容按照这样的层次组织：单词、句子、段落以及章。
- 代码也有类似的自底向上的组织层次：数据类型类似于单词，语句类似于句子，函数类似于段落，模块类似于章。
- 一个Python程序可以在不同文件中，这些文件是一个个模块。用import语句引入。
- import 模块名
- 模块名是另外一个Python文件的文件名,不包含扩展名,模块是可以运行的程序,
- “import 模块名”就是执行文件名为模块名的程序

# 引入模块中函数另一种方法

from 模块名 import *

这种方法引入模块中的所有函数，调用的时候不需要再加模块名

from 模块名 import 函数名

这种方法引入模块中的单个函数，调用的时候也不需要再加模块名

# 创建模块(两文件同一目录）

下面是triangle模块（文件名为 triangle.py）

```
import math
def area(a,b,c):
    s=(a+b+c)/2
    return (math.sqrt(s*(s-a)*(s-b)*(s-c)))
```

主程序是另一个文件（area.py）

```
import triangle
a=12
b=34
c=26
print(triangle.area(a,b,c))
```

# 模块名字空间

1. “主程序”指启动一个程序的代码。包含“主程序”的模块，它的模块空间就是全局名字空间
2. 模块空间中有:
   模块名，它是模块的文件名，
   但 包含“主程序”的模块名是”__main__”
   全局变量
   函数名
3. dir(模块名）：显示模块内容

# 包

- 我们已使用过单行代码、多行函数、独立程序以及同一目录下的多个模块。为了使Python应用更具可扩展性，你可以把多个模块文件组织成目录，称之为包。包是一个目录，其中包含一组模块文件和一个*init.py* 文件。如果模块存在于包中，使用“import 包名.模块名”形式导入包中模块，用以下形式调用函数：“包名.模块名.函数”
-

# sys模块

上面的area.py和triangle.py两个文件在同一个目录下，通过Python运行主程序area.py，会引用triangle模块，执行函数area。

模块的查找路径：sys.path

如不在同一目录，可用sys模块加入搜索路径后调用。

# sys模块中的常用量

| | |
|---|---|
| sys.argv | 从程序外部向程序传递参数 |
| sys.path | 模块搜索路径的字符串列表 |
| sys.stdin | 标准输入 |
| sys.stdout | 标准输出 |
| sys.stderr | 标准错误输出 |
| sys.getdefaultencoding() | 获取系统当前编码 |

# 命令行参数

- 如何不用input()函数向程序输入值？命令行参数可以解决这个问题
- sys.argv[0]　程序的文件名
- sys.argv[1]　第一个参数
- sys.argv[2]　第二个参数
- 。。。。。。
- sys.argv[n]　第n个参数
-

# 命令行参数例子

- import sys
- print(sys.argv[0])　#程序的文件名
- print(sys.argv[1])　#被乘数
- print(sys.argv[2])　#乘数
- print(int(sys.argv[1])*int(sys.argv[2]))

- 程序执行：
- c:\>python d:\mul.py 2 6
- d:\mul.py
- 2
- 6
- 12