Embedded Applications on Intelligent Systems

Fang Yuan (yfang@nju.edu.cn)

School of Electronic Science and Engineering
Nanjing University
Nanjing 210046

1. 模块化编程

Python 模块

模块化是程序设计的重要手段. 模块化有助于优化代码结构、减少错误、资源共享.

Python 强大的功能来源于丰富的第三方模块. Python 允许非常方便地创建新的模块, 使其具有灵活的扩展能力.

本章讨论:

- 如何使用已有模块
- 如何创建自己的模块

导入模块

以下是几种常见的导入模块的方法:

```
import math
print (math.sin(math.pi/6))
```

允许使用 math 模块里的所有符号, 但使用时需要加前缀 “math.”

函数dir()返回模块可导入符号的列表.

导入模块

```
import math as M
print (M.sin(M.pi/6))
```

将模块重新命名模块为 “M”. 这种用法主要目的是简化模块名称, 有时也用于处理兼容性问题:

```
import sys
if sys.version[0] == '3':
    import tkinter
elif sys.version[0] == '2':
    import Tkinter as tkinter
...
win = tkinter.Tk()
...
```

导入模块

仅导入的需要使用的符号, 使用时无需模块名前缀

```
from math import sin, cos, pi
print (sin(pi/6), cos(pi/6))

from math import sin as SIN
print (SIN(pi/6))
# error! since ``pi'' is not imported
```

导入模块

导入模块中除“_”开头的的所有符号, 且使用时无需模块名前缀

```
from math import *
print (sin(pi/6), exp(1))
```

这种方法一般不建议使用, 它容易导致符号的使用混乱. 例如树莓派 GPIO 模块 RPi.GPIO 模块中有一个函数 input(), 它与 Python 内建函数 input()重名.

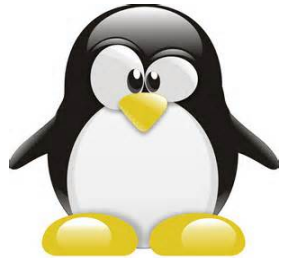
模块路径

允许导入的模块, 必须在特定的路径下. 指定这些路径的参数被保存在 sys 模块的 path 列表变量中:

```
>>> import sys
>>> print (sys.path)
```

通常, “当前目录” 包含在这个变量列表中. path 变量允许动态修改.

尝试在你的系统上观察有哪些目录允许导入模块



文件型模块

Python 源程序文件名本身就可以成为一个模块 (不含文件后缀 ".py"), 被其他程序导入.

例如, 当前目录下有两个文件,

```
$ ls
fibonacci.py    main.py
```

子模块 fibonacci.py

```
def fib(n):
    a, b = 0, 1
    while b < n:
        print (b, end=',␣')
        a, b = b, a + b

if __name__ == '__main__':
    fib(30)
```

子模块可以独立使用.

主模块 main.py

```
from fibonacci import fib
fib(20)
```

每个模块都有一个模块名变量__name__. 默认的模块名就是 Python 源文件不含后缀的文件名或目录名. 当它作为主程序运行时, 模块名是“__main__”.

语句if __name__ == '__main__':避免作为子模块被导入时被动运行其中的程序.

目录型模块

目录本身可以成为一个模块, 目录中的文件、子目录成为这个模块的子模块.

```
main.py                    (文件)
mymodule                   (目录)
      |
      `-- fibonacci.py     (文件)
      `-- __init__.py      (文件)
```

文件__init__.py可以是空文件, 用来标识这个目录是可以被导入的 Python 模块, 其中的语句会在导入时自动执行. 该文件通常用于模块的初始化.

模块的使用

Listing 1: main.py

```
from mymodule.fibonacci import fib
fib(20)
# 或
import mymodule.fibonacci as FB
FB.fib(20)
```

模块允许多层调用, 即: 若模块 A 导入了模块 B 的符号, 则当导入模块 A 时, 也可以直接从模块 A 中导入模块 B 的符号.