# Chapter II Tips



## 1 自省方法

在变量前后使用问号？，可以显示对象的信息：

```
b = [1, 2, 3]
#%%
b?
#%%
print?
```

结果如：

```
Signature: print(*args, sep=' ', end='\n', file=None, flush=False)
Docstring:
Prints the values to a stream, or to sys.stdout by default.

sep
  string inserted between values, default a space.
end
  string appended after the last value, default a newline.
file
  a file-like object (stream); defaults to the current sys.stdout.
flush
  whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

可以作为对象的自省，如果对象是一个函数or实例方法，定义过的文档字符串，也会显示出信息。

使用 ?? 可以显示函数的源码：

```
def add_numbers(a, b):
    return a + b

add_numbers?

add_numbers??
```

结果如下：

```
Signature: add_numbers(a, b)
Docstring: <no docstring>
Source:
def add_numbers(a, b):
    return a + b
File:      c:\users\vox1827\appdata\local\temp\ipykernel_24796\3064047894.py
Type:      function
```

还可以用于搜索IPython的命名空间，字符与通配符结合可以匹配所有的名字，例如如下代码，可以获取所有包含load的顶级Numpy命名空间：

```
import numpy as np
np.*load*?
################################
np.__loader__
np.load
np.loadtxt
```

# 2 % Command

可以使用`%run`命令运行所有的Python程序，文件中定义的变量（import、函数和全局变量，除非抛出异常），都可以在IPython shell中随后访问。

> 如果想让一个脚本访问IPython已经定义过的变量，可以使用`%run -i`

也可以使用`%load` ，它将脚本导入到一个代码格中。

使用`%paste` 和`%cpaste`函数，可以运行剪贴板中的代码。 — **已经用不了的**

# 3 Integrate Matplotlib

`%matplotlib`魔术函数配置了IPython shell和Jupyter notebook中的matplotlib，它创建的图不会出现或获取session的控制，直到结束，可以创建多个绘图窗口，而不会干扰控制台session：

```
# 运行%matplotlib可以进行设置，可以创建多个绘图窗口，而不会干扰控制台session
%matplotlib inline

import matplotlib.pyplot as plt
plt.plot(np.random.randn(50).cumsum())
```

# 4 变量和参数传递

赋值也被称作绑定，我们是把一个名字绑定给一个对象。变量名有时可能被称为绑定变量。

当你将对象作为参数传递给函数时，新的局域变量创建了对原始对象的引用，而不是复制。如果在函数里绑定一个新对象到一个变量，这个变动不会反映到上一层。因此可以改变可变参数的内容。

```
# 同一作用域
a = [1, 2, 3]
b = a
a.append(4)
b
# Result:
# [1, 2, 3, 4]


# 参数传递到局域
def append_element(some_list, element):
    some_list.append(element)

data = [1, 2, 3]
append_element(data, 4)
data
```

```
# Result:
# [1, 2, 3, 4]
```

# 5 动态引用&强类型

Python是强类型化语言，每个对象都有明确的类型，默许转换只会发生在特定的情况下。可以用`isinstance`函数检查对象是某个类型的实例。该函数也可以使用类型元组，检查对象的类型是否在元组中。

```
a = 5
isinstance(a, int)
# Result:
# True

# 使用类型元组
a = 5; b = 4.5
print(isinstance(a, (int, float)))
print(isinstance(b, (int, float)))
# Result
# True
# True
```

# 6 可变与不可变对象

Python中的大多数对象，比如列表、字典、NumPy数组，和用户定义的类型，都是可变的，这些对象或包含的值可以被修改

其他的，例如**字符串和元组，是不可变的**