

第1章 Python语言概述

1.1 Python是这样一种语言

- Python是一门跨平台、开源、免费的解释型高级动态编程语言。
- Python支持命令式编程（How to do）、函数式编程（What to do），完全支持面向对象程序设计，拥有大量扩展库。
- 胶水语言：可以把多种不同语言编写的程序融合到一起实现无缝拼接，更好地发挥不同语言和工具的优势，满足不同应用领域的需求。

1.1 Python是这样一种语言

- 问题解决：把列表中的所有数字都加5，得到新列表。（命令式编程）

```
>>> x = list(range(10))
```

创建列表

```
>>> x
```

空列表

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> y = []
```

```
>>> for num in x:
```

循环，遍历x中的每个元素

```
    y.append(num+5)
```

列表方法，在尾部追加元素

```
>>> y
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
>>> [num+5 for num in x]
```

列表推导式

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

1.1 Python是这样一种语言

- 问题解决：把列表中的所有数字都加5，得到新列表。（函数式编程）

```
>>> x = list(range(10))
```

```
>>> x
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> def add5(num):  
    return num+5
```

定义函数，接收一个数字，加5后返回

```
>>> list(map(add5, x))
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
>>> list(map(lambda num: num+5, x))
```

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

把函数add5映射到x中的每个元素

lambda表达式，等价于函数add5

1.2 Python版本之争

- Python目前存在2.x和3.x两个系列的版本，互相之间**不兼容**。
- 在选择Python版本的时候，一定要先考虑清楚自己学习Python的目的是什么，打算做哪方面的开发，该领域或方向有哪些扩展库可用，这些扩展库最高支持哪个版本的Python。这些问题全部确定以后，再最终确定选择哪个版本。

1.3 Python编程规范与代码优化建议

(1) 缩进

- ✓ 类定义、函数定义、选择结构、循环结构、with块，行尾的冒号表示缩进的开始。
- ✓ python程序是依靠代码块的缩进来体现代码之间的逻辑关系的，缩进结束就表示一个代码块结束了。
- ✓ 同一个级别的代码块的缩进量必须相同。
- ✓ 一般而言，以4个空格为基本缩进单位。

```
with open(fn) as fp:
    for line in csv.reader(fp):
        if line:
            print(*line)
```

1.3 Python编程规范与代码优化建议

(2) 每个import语句只导入一个模块，最好按标准库、扩展库、自定义库的顺序依次导入。

```
import csv
import random
import datetime
import pandas as pd
import matplotlib.pyplot as plt
```

1.3 Python编程规范与代码优化建议

(3) 最好在每个类、函数定义和一段完整的功能代码之后增加一个空行，在运算符两侧各增加一个空格，逗号后面增加一个空格。

```
sockDianming.close()
sockDianming = None

# 开始点名
def buttonStartDianmingClick():
    # 如果还没有注册，拒绝运行
    if int_zhuce.get() == 0:
        tkinter.messagebox.showerror('很抱歉', '请联系作者进行软件注册!')
        return
    if int_zuoye.get() == 1:
        tkinter.messagebox.showerror('很抱歉', '现在正在收作业')
        return
    if int_canDianming.get() == 1:
        tkinter.messagebox.showerror('很抱歉', '现在正在点名')
        return
    tkinter.messagebox.showinfo('恭喜', '设置成功，现在开始点名')
    #开始点名
    int_canDianming.set(1)
    global tDianming_id
    t = threading.Thread(target=thread_Dianming)
    t.start()
    tDianming_id = t.ident
    buttonStartDianming = tkinter.Button(root, text='开始点名', command=buttonStartD
    buttonStartDianming.place(x=20, y=60, height=30, width=100)

def buttonStopDianmingClick():
    # 如果还没有注册，拒绝运行
    if int_zhuce.get() == 0:
        tkinter.messagebox.showerror('很抱歉', '请联系作者进行软件注册!')
```

Diagram illustrating code formatting suggestions:

- Red arrows pointing to empty lines after function definitions and code blocks, labeled "空行" (Empty line).
- Green arrows pointing to spaces around operators and after commas, labeled "有空格" (With space) and "没有空格" (Without space).

1.3 Python编程规范与代码优化建议

(4) 尽量不要写过长的语句。如果语句过长，可以考虑拆分成多个短一些的语句，以保证代码具有较好的可读性。如果语句确实太长而超过屏幕宽度，最好使用续行符（line continuation character）“\”，或者使用圆括号将多行代码括起来表示是一条语句。

```
x = 1 + 2 + 3\  
    + 4 + 5\  
    + 6
```

```
y = (1 + 2 + 3  
     + 4 + 5  
     + 6)
```

1.3 Python编程规范与代码优化建议

(5) 虽然Python运算符有明确的优先级，但对于复杂的表达式建议在适当的位置使用**括号**使得各种运算的隶属关系和顺序更加明确、清晰。

1.3 Python编程规范与代码优化建议

(6) 注释

- ✓ 以符号#开始，表示本行#之后的内容为注释。
- ✓ 包含在一对三引号'''...'''或"""..."""之间且不属于任何语句的内容将被解释器认为是注释。

```
# 读取并显示上面代码生成的csv文件内容
with open(fn) as fp:
    for line in csv.reader(fp):
        if line:
            print(*line)
```

1.3 Python编程规范与代码优化建议

(7) 在开发速度和运行速度之间尽量取得最佳平衡。

- ✓ 内置对象运行速度最快，标准库对象次之，用C或Fortran编写的扩展库速度也比较快，而纯Python的扩展库往往速度慢一些。
- ✓ 在开发项目时，应优先使用Python内置对象，其次考虑使用Python标准库提供的对象，最后考虑使用第三方扩展库。

1.3 Python编程规范与代码优化建议

(8) 根据运算特点选择最合适的数据类型来提高程序的运行效率。

- ✓ 如果定义一些数据只是用来频繁遍历并且关心顺序，最好优先考虑元组。
- ✓ 如果需要频繁地测试一个元素是否存在于一个序列中并且不关心其顺序，尽量采用集合。
- ✓ 列表和元组的in操作的时间复杂度是线性的，而对于集合和字典却是常数级的，与问题规模几乎无关。

1.3 Python编程规范与代码优化建议

- （9）充分利用关系运算符以及逻辑运算符and和or的惰性求值特点，合理组织条件表达式中多个条件的先后顺序，减少不必要的计算。
- （10）充分利用生成器对象或类似迭代对象的惰性计算特点，尽量避免将其转换为列表、元组等类型，这样可以减少对内存的占用，降低空间复杂度。
- （11）减少内循环中的无关计算，尽量往外层提取。

1.3 Python编程规范与代码优化建议

```
>>> import this
The Zen of Python, by Tim Peters
```

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
```

```
Readability counts.
```

```
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

1.3 Python编程规范与代码优化建议

- 以动手实践为荣，以只看不练为耻。
- 以打印日志为荣，以单步跟踪为耻。
- 以空格分隔为荣，以制表分隔为耻。
- 以单元测试为荣，以手工测试为耻。
- 以代码重用为荣，以复制粘贴为耻。
- 以多态应用为荣，以分支判断为耻。
- 以Pythonic为荣，以冗余拖沓为耻。
- 以总结思考为荣，以不求甚解为耻。

1.4 Anaconda3开发环境的安装与使用

- 默认编程环境： **IDLE**
- 其他常用开发环境：
 - **Eclipse+PyDev**
 - **pyCharm**
 - wingIDE
 - **Eric**
 - PythonWin
 - **Anaconda3**（内含Jupyter和Spyder）：<https://www.anaconda.com/download>
 - zwPython

Python安装

- 官网下载地址: <https://www.python.org/downloads/>
 - 32位Windows系统下载 Windows x86 executable installer
 - 64位Windows系统下载 Windows x86-64 executable installer
 - macOS系统默认自带Python v2.x (需要手动安装Python 3)

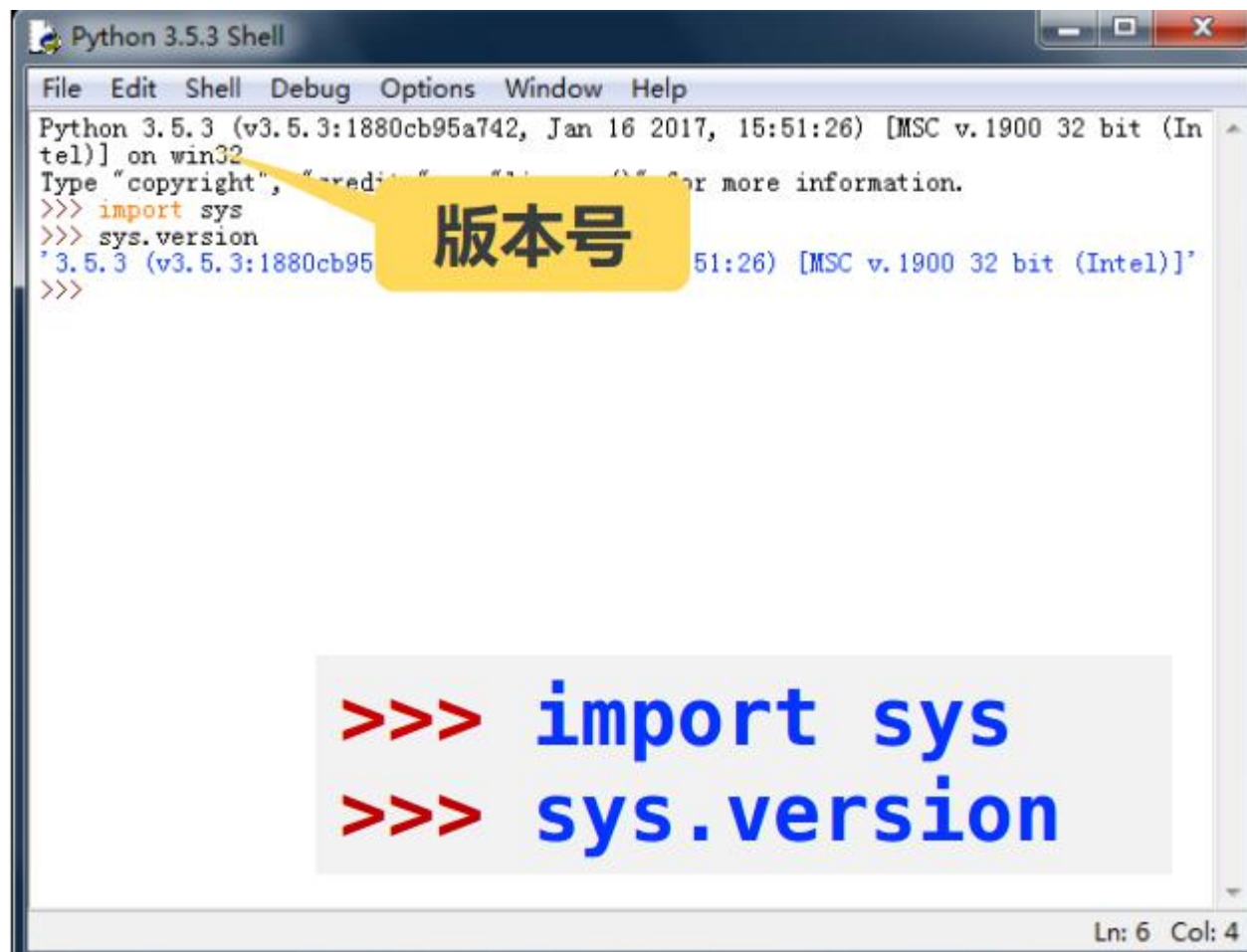
请官网下载 并安装python-3.6.x-macosx10.6.pkg (如果需要支持中文输入, 请先安装ActiveTcl8.5.18.x-macosx10.5-i386-x86_64-threaded.dmg)。如果安装了Homebrew, 直接通过brew install python3 --with-tcl-tk安装即可 (个别版本会造成IDLE崩溃)。



Python安装



IDLE基本使用简介



The screenshot shows a 'Python 3.5.3 Shell' window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The command prompt shows the following text:

```
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 15:51:26) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits()" or "help()" for more information.
>>> import sys
>>> sys.version
'3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 15:51:26) [MSC v.1900 32 bit (Intel)]'
>>>
```

A yellow callout box with the text '版本号' (Version Number) points to the output of the `sys.version` command.

Below the screenshot, a grey box contains the following code snippets:

```
>>> import sys
>>> sys.version
```

The status bar at the bottom right of the window indicates 'Ln: 6 Col: 4'.

Python运行方式：Shell方式（交互式环境）

- 直接在提示符“>>>”后面输入相应的命令（代码）并回车会立刻执行
- IDLE交互界面可以当做简便计算器来使用
- 多尝试在交互式环境中输入一条指令，检查它是否能工作
- 执行顺利，可以马上看到执行结果
- 执行出错，会抛出异常，不用担心

```
>>> 3+5
8
>>> 3*(2+6)                # 注意乘法使用星号*
24
>>> import math             # 数学标准库
>>> math.sqrt(9)
3.0
>>>
```

遇到错误怎么办

- 如何寻求帮助？
- 独自解决编程问题可能比你想的要容易

```
>>> 2/0
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    2/0
ZeroDivisionError: division by zero
>>>
```

Traceback 部分显示
遇到异常的
特定指令和行号

- 可以在网上搜索错误信息

具体的错误信息

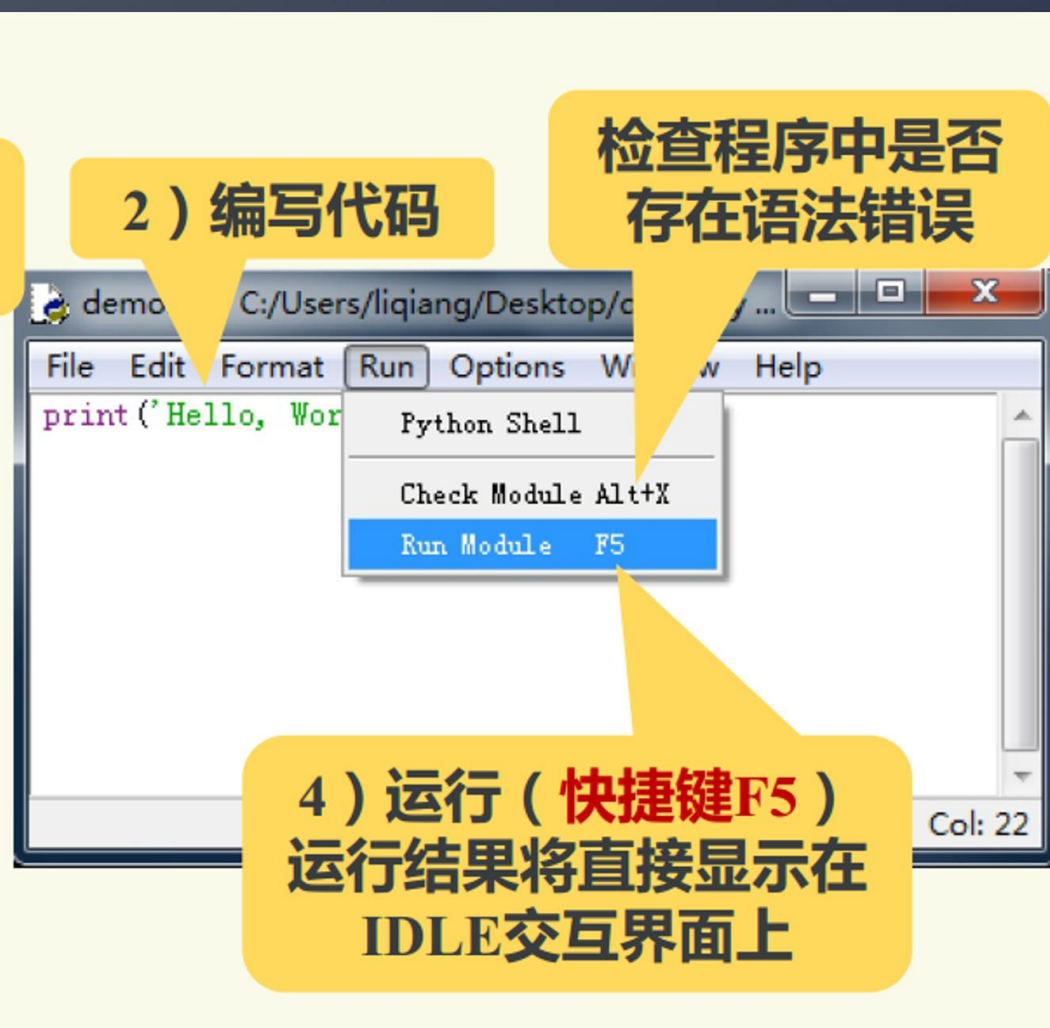
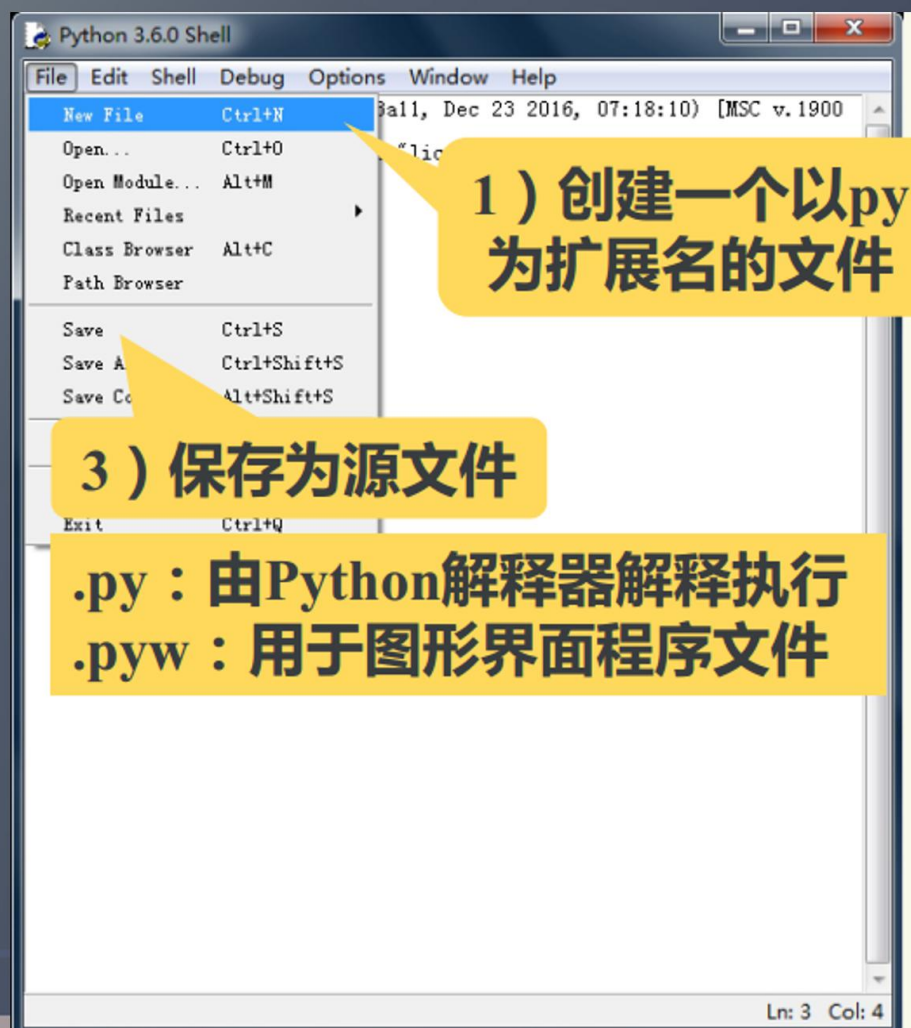
第一个 Hello World! 程序代码

```
>>> print('Hello World!')  
Hello World!  
>>>
```

- 第一行：>>> 是Python语言运行环境的提示符
- 第二行：Python语句的执行结果
- 初次接触print()函数：要让Python打印出指定的文字，使用print()函数
- 希望打印的文字一般用单引号或者双引号括起来，但不能混用。括起来的文本叫字符串

Python运行方式：文件方式

- 针对 Python 的编辑器（无代码合并，但有语法标签高亮和代码自动完成功能）、类浏览器和调试器





python

Python程序实例解析

- “温度转换”问题分析
- “温度转换”实例编写

“温度转换”问题分析

输入：带华氏或摄氏标志的温度值

处理：根据温度标志选择适当的温度转换算法

输出：带摄氏或华氏标志的温度值

根据华氏和摄氏温度定义，利用转换公式如：

$$- C = (F - 32) / 1.8$$

$$- F = C * 1.8 + 32$$

其中， C表示摄氏温度， F表示华氏温度

“温度转换”实例编写

```
#TempConvert.py
TempStr = input("请输入带有符号的温度值：")
if TempStr[-1] in ['F', 'f']:
    C = (eval(TempStr[0:-1]) - 32)/1.8
    print("转换后的温度是{:.2f}C".format(C))
elif TempStr[-1] in ['C', 'c']:
    F = 1.8*eval(TempStr[0:-1]) + 32
    print("转换后的温度是{:.2f}F".format(F))
else:
    print("输入格式错误")
```

```
请输入带有符号的温度值：45C
转换后的温度是113.00F
>>>
```

1.5 安装扩展库的几种方法

- pip在线安装（命令提示符环境，建议切换至Python安装目录中的scripts文件夹执行）
- pip离线安装：<https://www.lfd.uci.edu/~gohlke/pythonlibs/>
- exe安装，不是每个扩展库都支持
- conda在线安装
- 如果机器上安装了多个Python开发环境，那么在一个环境下安装的扩展库无法在另一个环境下使用，需要分别安装。

1.5 安装扩展库的几种方法

pip命令示例	说明
<code>pip download SomePackage[==version]</code>	下载扩展库的指定版本，不安装
<code>pip freeze [> requirements.txt]</code>	以requirements的格式列出已安装模块
<code>pip list</code>	列出当前已安装的所有模块
<code>pip install SomePackage[==version]</code>	在线安装SomePackage模块的指定版本
<code>pip install SomePackage.whl</code>	通过whl文件离线安装扩展库
<code>pip install package1 package2 ...</code>	依次（在线）安装package1、package2等扩展模块
<code>pip install -r requirements.txt</code>	安装requirements.txt文件中指定的扩展库
<code>pip install --upgrade SomePackage</code>	升级SomePackage模块
<code>pip uninstall SomePackage[==version]</code>	卸载SomePackage模块的指定版本

把SomePackage替换
为实际要安装或卸载
的扩展库名

<http://www.lfd.uci.edu/~gohlke/pythonlibs/>
下载时选择合适版本，并且不要修改文件名

1.6 标准库与扩展库对象的导入与使用

- Python默认安装仅包含基本或核心模块，启动时也仅加载了基本模块，在需要时再显式地导入和加载标准库和第三方扩展库（需正确安装），这样可以减小程序运行的压力，并且具有很强的可扩展性。
- 从“木桶原理”的角度来看，这样的设计与安全配置时遵循的“最小权限”原则的思想是一致的，也有助于提高系统安全性。

模块导入与使用

使用方式一

- 导入模块
 - ✓ `import 模块名 [as 别名]`
- 调用函数
 - ✓ `模块名.对象名`
 - ✓ `别名.对象名`

使用方式二

- 导入模块的函数
 - ✓ `from 模块名 import 对象名 [as 别名]`
- 调用函数
 - ✓ `对象名`

1.6.1 import 模块名 [as 别名]

```
>>> import math                                #导入标准库math
>>> math.sin(0.5)                               #求0.5（单位是弧度）的正弦
0.479425538604203
>>> import random                              #导入标准库random
>>> n = random.random()                        #获得[0,1) 内的随机小数
>>> n = random.randint(1,100)                  #获得[1,100]区间上的随机整数
>>> n = random.randrange(1, 100)               #返回[1, 100)区间中的随机整数
>>> import os.path as path                     #导入标准库os.path，并设置别名为path
>>> path.isfile(r'C:\windows\notepad.exe')
True
>>> import numpy as np                         #导入扩展库numpy，并设置别名为np
>>> a = np.array((1,2,3,4))                    #通过模块的别名来访问其中的对象
>>> a
array([1, 2, 3, 4])
>>> print(a)
[1 2 3 4]
```


1.6.2 from 模块名 import 对象名[as 别名]

```
>>> from math import sin
```

#只导入模块中的指定对象，访问速度略快

```
>>> sin(3)
```

```
0.1411200080598672
```

```
>>> from math import sin as f
```

#给导入的对象起个别名

```
>>> f(3)
```

```
0.1411200080598672
```

1.6.3 from 模块名 import *

>>> from math import *	#导入标准库math中所有对象
>>> sin(3)	#求正弦值
0.1411200080598672	
>>> gcd(36, 18)	#最大公约数
18	
>>> pi	#常数 π
3.141592653589793	
>>> e	#常数e
2.718281828459045	
>>> log2(8)	#计算以2为底的对数值
3.0	
>>> log10(100)	#计算以10为底的对数值
2.0	
>>> radians(180)	#把角度转换为弧度
3.141592653589793	