

# Python程序设计



# 潘浩源

联系信息:

邮件 [hypan@szu.edu.cn](mailto:hypan@szu.edu.cn)

QQ群

Blackboard

个人研究: Wireless+AI

5G/6G无线通信与网络、物联网、无线系统设计、  
软件无线电



群名称: 2022 Python程序设计  
群 号: 860830989

# 课程内容

## 基础

- (1) Python基础知识
- (2) 数据结构
- (3) 字符串
- (4) 函数与函数式编程
- (5) 面向对象编程
- (6) Python标准库
- (7) 错误、调试和测试
- (8) IPython

## 实战

- (1) 交互式图形编程
- (2) 科学计算
- (3) 数据分析
- (4) 图像处理
- (5) 网络编程
- (6) 根据课程进度而定



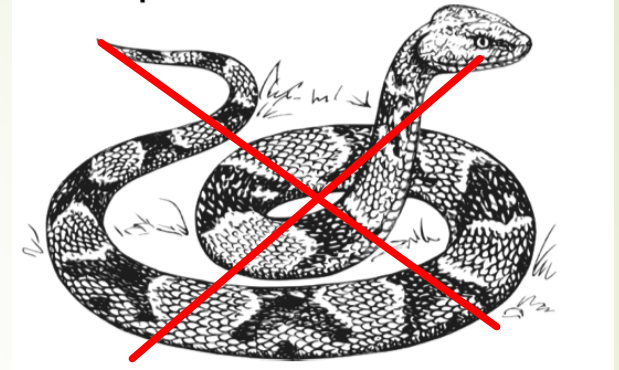
# Python是一种语言

有赞:

<https://www.bilibili.com/video/av971490872>

有弹

<https://www.bilibili.com/video/BV1gV411b7Rg>



So how does one learn how to code?

Think of it like learning a new language

The only way to learn is to try “listen” and “speak” the language

see others code

study example solutions

write your own codes!

# 课程内容

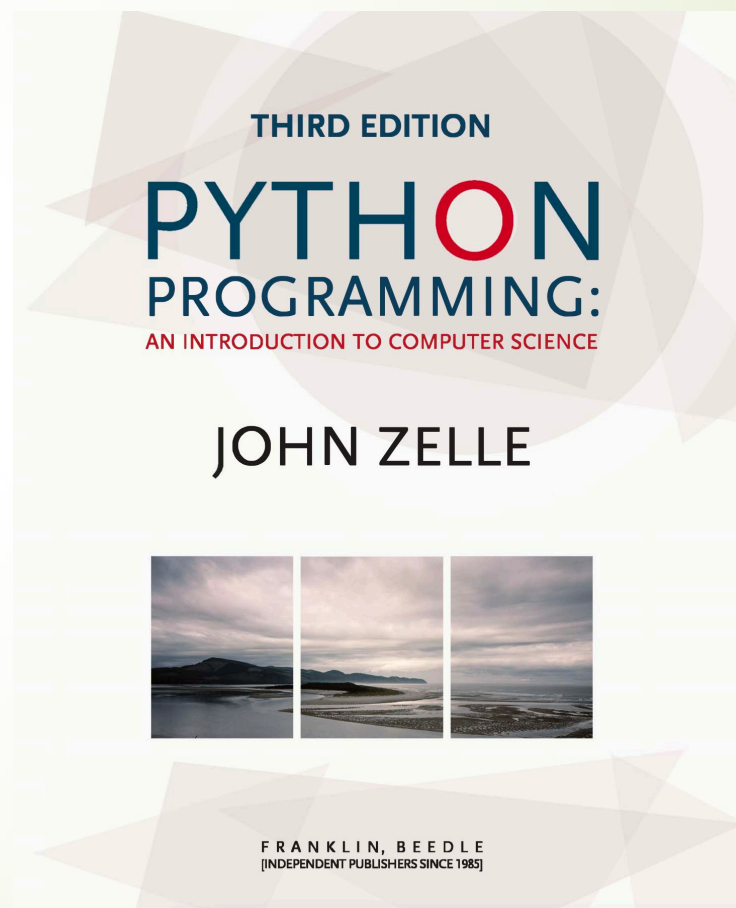
理论课（星期五 第7-8节）

- 讲解课程内容
- Python语法、结构、实例...
- 要求出席

实验课（星期五 第9-10节）

- 编程练习/展示
- 助教答疑（助教：杨帅）
- 要求出席

# 参考书



# 参考资料

©2012-2015 - Laurent Pointal Memento v2.0.6  
License Creative Commons Attribution 4 简体中文译: 心蛛, 2018

## Python 3 速查卡

最新版参见:  
<https://perso.limsi.fr/pointal/python:memento>

integer, float, boolean, string, bytes

### 基础数据类型

```
int 783 0 -192 0b010 0o642 0xF3
      zero      binary octal  hexa
float 9.23 0.0 -1.7e-6
bool True False
str "One\nTwo"
    转义字符: 换行
    'I\m'
    转义字符 '
bytes b"toto\xfe\775"
      十六进制 八进制 不可易对象
```

多行文本:

```
"""X\tY\tZ
1\t2\t3"""
```

转义字符: 跳格

■ 有序序列, 快速索引访问, 值可重复

```
list [1, 5, 9] ["x", 11, 8.9] ["mot"]
tuple (1, 5, 9) 11, "y", 7.4 ("mot",)
值不能修改 (不可易的, immutables) 用逗号分隔的表达式 → tuple
str bytes (有序的字符 / 字节序列)
```

■ 关键字容器, 未预设顺序, 用键来快速访问, 每个键是唯一的

```
字典 dict {"key": "value"} dict(a=3, b=4, k="v")
      (键 / 值关联) {1: "one", 3: "three", 2: "two", 3.14: "pi"}
集合 set {"key1", "key2"} {1, 9, 3, 0} set
键 = 可哈希化的值 (基础数据类型, 不可易对象……) frozenset 不可易的集, 冻结集
```

### 容器类型

用于表示变量、函数、模块、类……的名字

### 标识符

**a...zA...Z\_** 紧跟着 **a...zA...Z\_0...9**  
□ 读音符号及汉字都合法, 但应避免使用  
□ 禁止使用语言关键字  
□ 字母大小写应该便于区分  
◎ **a toto x7 y\_max BigOne**  
◎ **8y and for**

=

### 变量赋值

■ 赋值 ⇔ 把值与一个名字进行绑定  
1) 计算表达式右侧的值  
2) 按顺序对左侧的名字赋值  
**x=1.2+8+sin(y)**  
**a=b=c=0** 赋给同一个值  
**y, z, r=9.2, -7.6, 0** 多个赋值  
**a, b=b, a** 值交换  
**a, \*b=seq**  
**\*a, b=seq** 将序列拆分成条目和列表  
**x+=3** 自增 ⇔ **x=x+3**  
**x-=2** 自减 ⇔ **x=x-2**  
**x=None** «未定义» 常量  
**del x** 删除名字 **x**

**int("15") → 15**

**int("3f", 16) → 63**

**int(15.56) → 15**

**float("-11.24e8") → -1124000000.0**

**round(15.56, 1) → 15.6** 舍入到小数点1位 (0位 → 整数)

**bool(x)** **False** 对应空 **x**, 空容器 **x**, **None** 或 **False x**; **True** 对应其他的 **x**

**str(x) → "..."** **x** 所对应的字符串表示 (参见背面的格式化内容)

**chr(64) → '@'** **ord('@') → 64** 代码 ↔ 字符

**repr(x) → "..."** **x** 对应的直接表达的 (literal) 字符串

**bytes([72, 9, 64]) → b'H\t@'**

**list("abc") → ['a', 'b', 'c']**

**dict([(3, "three"), (1, "one")]) → {1: 'one', 3: 'three'}**

**set(["one", "two"]) → {'one', 'two'}**

分隔字符串 **str** 与 **str** 序列 → 组装的 **str**

**':'.join(['toto', '12', 'pswd']) → 'toto:12:pswd'**

**str** 用空格分割 → 由 **str** 构成的 **list**

**"words with spaces".split() → ['words', 'with', 'spaces']**

**str** 以 **str** 字符串来分割 → 由 **str** 构成的 **list**

**"1,4,8,2".split(",") → ['1', '4', '8', '2']**

同一种类型的序列 → 另一种类型的 **list** (通过列表推导式)

**[int(x) for x in ('1', '29', '-3')] → [1, 29, -3]**

**type(表达式)**

可以用第二个参数设置整数的基数

截断小数部分

舍入到小数点1位 (0位 → 整数)

### 类型转换

# 在线课程

1 Python语言程序设计，北京理工大学，  
<https://www.icourse163.org/course/BIT-268001>

2（**强烈推荐，特别是非计算机专业学生**）

edX， MIT 6.001， Introduction to Computer Science and  
Programming Using Python

(或通过

MITOPENCOURSEWARE <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/>)



# 在线课程

不管用什么语言编程，官方文档永远是学习最好的选择，最新最详细的代码说明都在官方文档里。

Python也是这样，虽然市场上的教程多如牛毛，但官方文档必须要看的。

- ➡ Official Python documentation

<http://docs.python.org/>

- ➡ Official Python tutorial.

<http://docs.python.org/tutorial/>

# 课程考核

实验成绩	实验代码+实验报告	50%
平时成绩	签到、作业、小测、阶段测验...	25%
期末大作业	期末综合型大作业，根据作业演示情况、PPT讲解、大作业报告打分	25%

# 作业政策

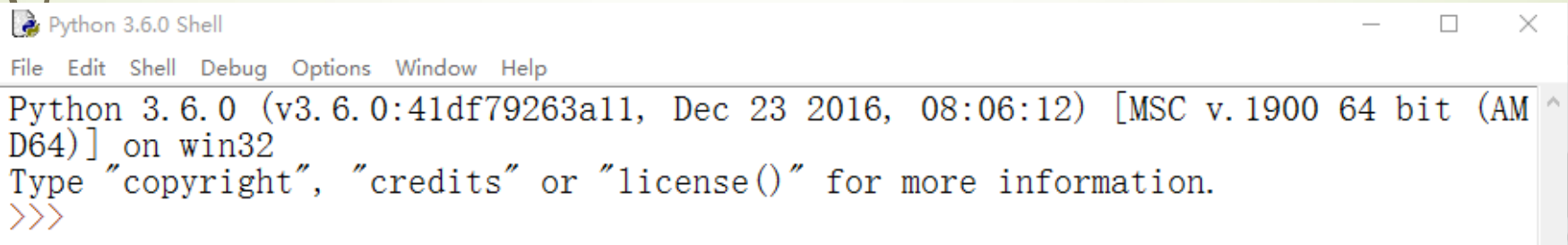
## 合作？

- **Okay**
  - 互相帮助调试程序
  - 讨论解决问题的方法
  - 在提交时提供所有“合作者”的姓名
- **Not Okay**
  - 复制代码（来自班级同学的或网上的代码）
  - 一起写代码（在线合作？**No!**）
  - 向他人显示/发送代码

# 实验 0 （今天）

**安装Python:** <https://www.bilibili.com/video/BV1N7411t7RJ>

- Windows: 安装Python默认编程环境IDLE，或者一些开发环境，如anaconda（**建议Python 3**）
- 安装后，进入Python运行界面
- 熟悉Python编辑、开发、和运行环境
- 输出一些信息，例如 `print(“Hello, Python”)`
- 打印你的学号和名字



```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```



# Q&A



# Python程序设计

## 第1课 Python基础知识1

# Python是一种怎样的语言

- ⑩ Python是一门跨平台、开源、免费的**解释型高级动态编程语言**。
- ⑩ Python支持**命令式编程、函数式编程**，面向**对象程序设计**，语法简洁清晰，拥有大量的**成熟的扩展库**。
- ⑩ **胶水语言**：可以把多种不同语言编写的程序融合到一起，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。

# Python的哲学：优雅，明确，简单

➤ From The Zen of Python

➤ (Python之禅 <https://www.python.org/dev/peps/pep-0020/>)

- 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!



# Python 设计目标

- 易于学习和使用:
  - 自动内存管理
  - 高级内置数据结构
  - 大型标准库
- 可读性:
  - 语法直观，简单
- 动态行为:
  - 解释语言
  - 动态类型 (dynamic type)
- 可扩展性:
  - 可重用代码：模块和包
  - 开源
  - 易于用C编写新模块

2个主要版本：Python 2和Python 3； Python 3在设计时，没有考虑向下兼容

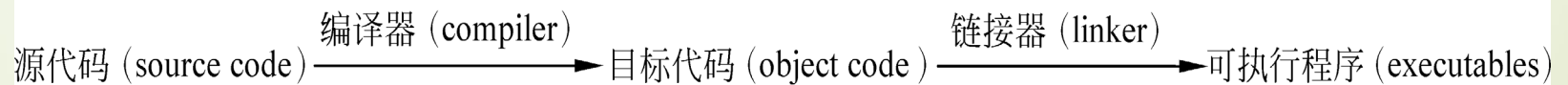
# Python如何提高生产力

“ One Python programmer can finish in two months what two C++ programmers cannot complete in a year” [Guido van Rossum, ‘Comparing Python to Other Languages’ , 1997 (online)]

- Interpreted Language: No compilation step.

将源文件转换成机器语言有以下两种转换方法

编译：编译器（**Compiler**）将源代码翻译成目标语言



解释：解释器（**Interpreter**）直接解释执行高级程序设计语言

# Python如何提高生产力

“ One Python programmer can finish in two months what two C++ programmers cannot complete in a year” [Guido van Rossum, ‘Comparing Python to Other Languages’ , 1997 (online)]

- Interpreted Language: No compilation step.
- Debugging is easy:
  - introspection abilities (state of the interpreter is accessible during program runtime). 自省能力（在程序运行时可以访问解释器的状态）。
  - read-eval-print loop (REPL).
  - elaborate error handling system.
- Faster development times due to more compact code.
  - typically 3-5 times shorter than Java code.
  - typically 5-10 times shorter than C++ code.



# Python用在哪？

- **Application development.**

(from small command line tools, to large GUI based applications and 3D games).

- **Web development.**

(Yelp, YouTube, Reddit, WordsEye, ... )

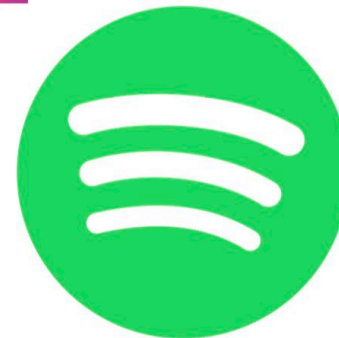
- **Scientific/numeric computing.**

(machine learning, physics, bioinformatics, NLP,...)

# Python的广告无处不在

Will Python help me get a job / help me in the real world?

Python in Business



Mount & Blade

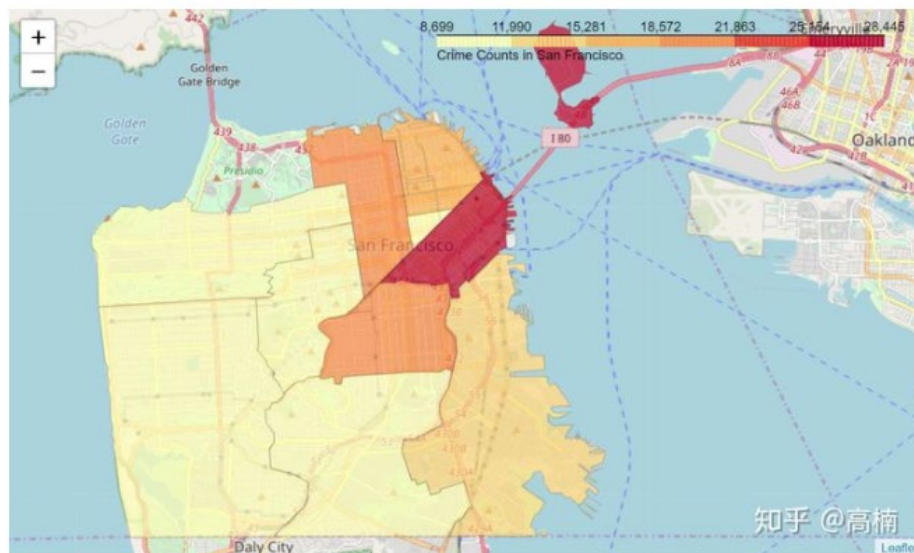


# Python的广告无处不在

## Python 如何画出漂亮的地图?

### 9. 创建Choropleth Map (颜色深浅代表各区犯罪事件数目)

```
m = folium.Map(location=[37.77, -122.4], zoom_start=12)
folium.Choropleth(
    geo_data=san_geo,
    data=disdata,
    columns=['Neighborhood', 'Count'],
    key_on='feature.properties.DISTRICT',
    #fill_color='red',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    highlight=True,
    legend_name='Crime Counts in San Francisco'
).add_to(m)
m
```





# Python的广告无处不在

## 在设计领域python可以做什么？

本人是一名产品设计师，最近在自学python。也许是现在学的太浅，看不出来python对今后的设计事业有什么帮助，所以想请大神指点迷津。显示全部

关注问题

写回答

邀请回答

好问题 5

添加评论

分享

...

查看全部 5 个回答



LucasX

谢邀，人在火星，刚下飞船。外星人太多，匿了。

40 人赞同了该回答

@微软亚洲研究院 前段时间刚出了一篇Paper，让AI自动化排版过程。当然，看懂算法了你可以用Python实现。



FASHION  
ICONOS FASHION  
NATIONAL GRAPHICS JAN  
MEZCLAS DE ESTILISTA  
BLAZER CAMISETA JERSEY  
BIQUINI  
CAMISA BLANCE ALTA CON  
SENSATO  
RELLENO DE LABIOS Y PAST  
ILASTARA

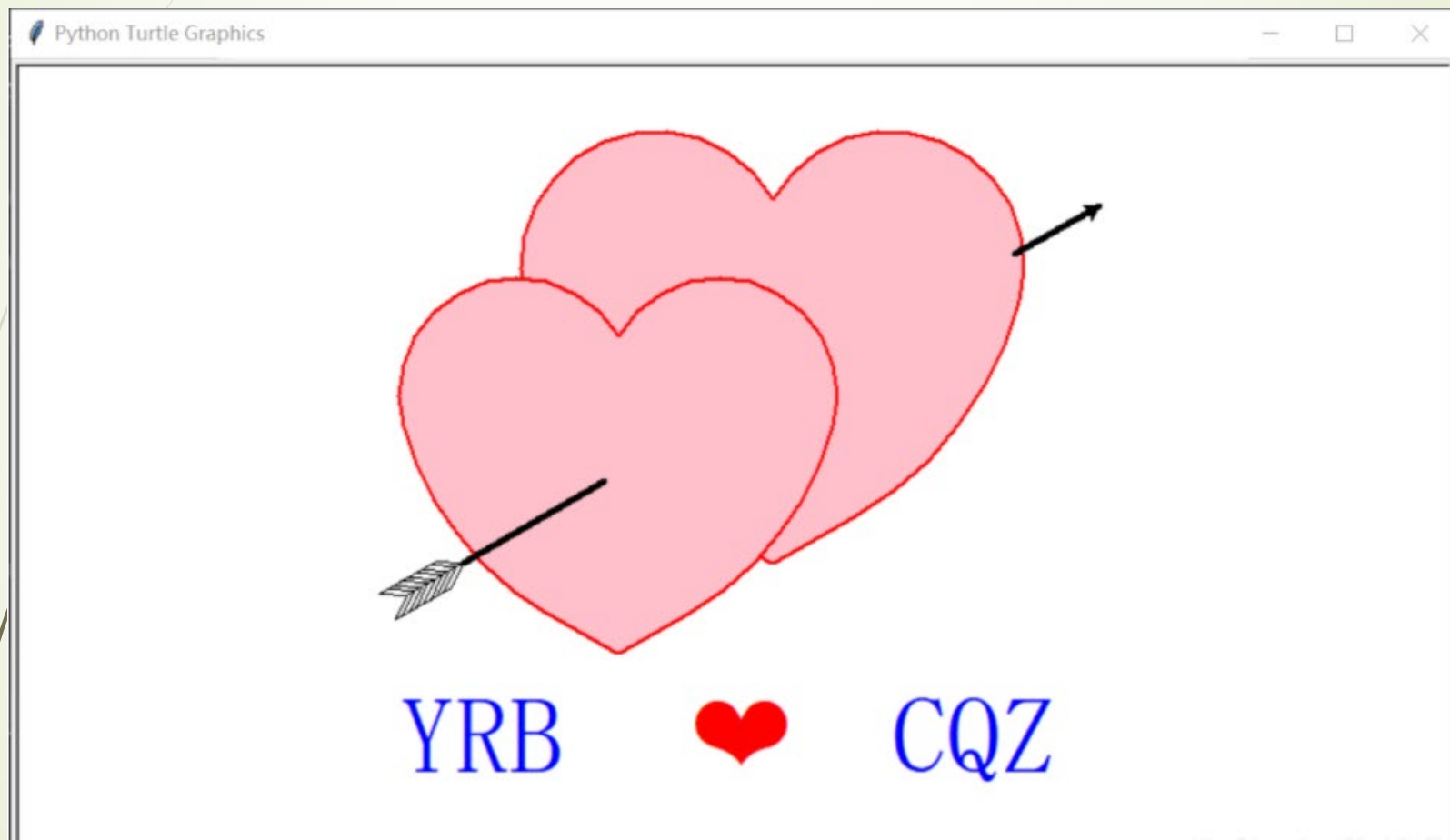


(a) visual-textual presentation layout



(b) real magazine cover

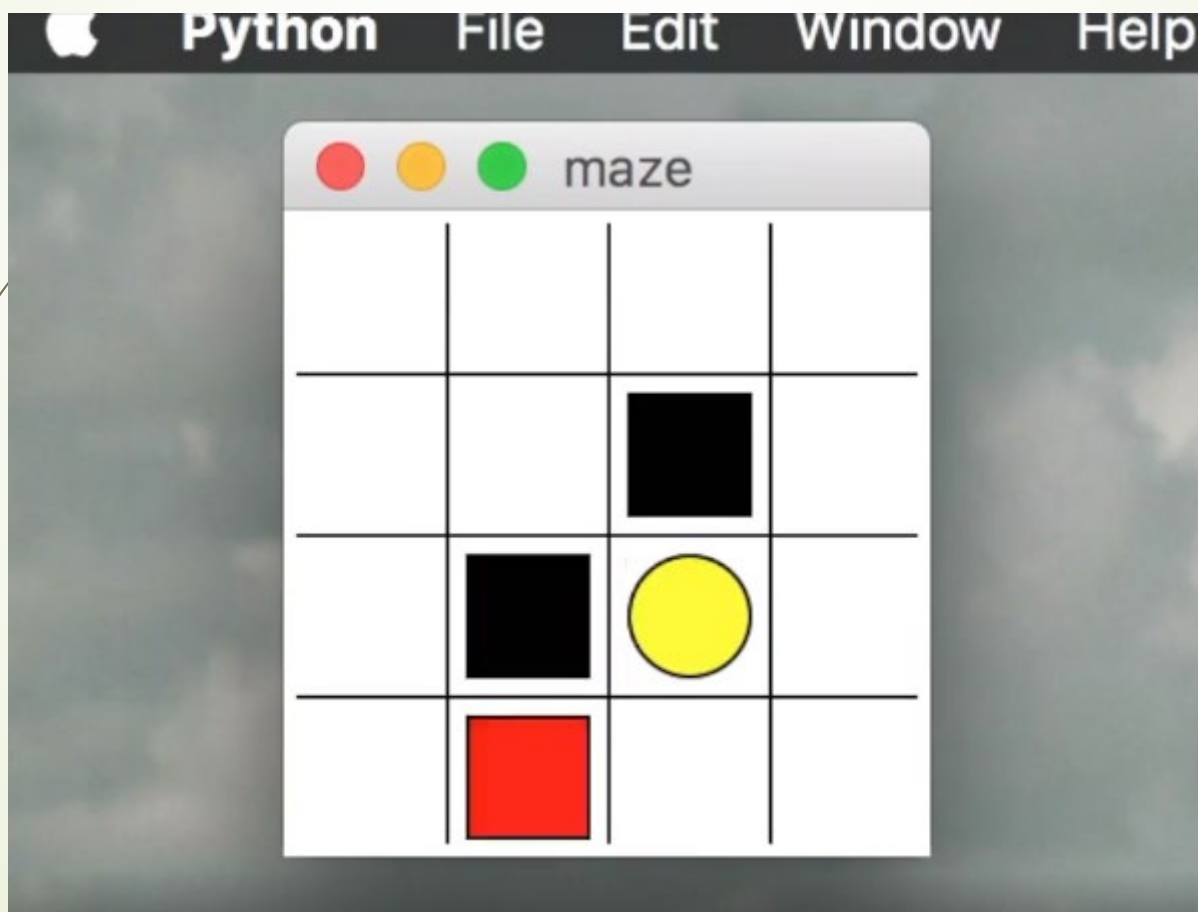
# Python也给生活带来小情趣





# Python与科研

机器学习（强化学习例子）



# Python与科研

无线系统:

这居然是个简单的Wi-Fi发送机? !

```
1  from gnuradio import gr
2  from gnuradio import eng_notation
3  from gnuradio.eng_option import eng_option
4  from gnuradio import digital
5  from optparse import OptionParser
6
7  from uhd_interface import uhd_transmitter
8  import transmit_path
9
10 import time, sys, struct, random
11
12
13 class my_top_block(gr.top_block):
14     def __init__(self, options):
15         gr.top_block.__init__(self)
16         if options.freq is not None:
17             self.d_tx_enable = True
18             u = uhd_transmitter(options.tx_args,
19                                options.bandwidth,
20                                options.tx_freq, options.tx_gain,
21                                options.spec, options.antenna,
22                                options.verbose, options.external)
23         elif options.outfile is not None:
24             self.d_tx_enable = False
25             u = gr.file_sink(gr.sizeof_gr_complex, options.outfile)
26         else:
27             raise SystemExit("--freq or --outfile must be specified\n")
28
29         tx = transmit_path.ofdm_transmit_path(options)
30         self.connect(tx, u)
31
32         self.tx = tx
33         self.u = u
34
35     def send_pkt(self, payload='', time=None, eof=False, msgq_tx_enable=False):
36         if eof and not msgq_tx_enable:
37             self.tx.send_eof()
38         elif eof and msgq_tx_enable:
39             self.tx.send_eof2()
40
41         if self.d_tx_enable:
42             if msgq_tx_enable:
43                 self.tx.send_samples(payload, time)
44             else:
45                 self.tx.send_pkt(payload, time)
46
```



有弹：

<https://www.bilibili.com/video/BV1gV411b7Rg>

不足：

1. python是一种解释型编程语言，因此大部分的py代码执行速度要比编译型语言（比如C++和java）慢的多。
2. **不容易维护：**因为Python是一种动态类型语言，所以根据上下文，同样的事情可能很容易意味着不同的东西。随着Python应用程序变得越来越大，越来越复杂，这可能难以维护，因为错误将难以追踪和修复，因此需要经验和洞察才能知道如何设计代码或编写单元测试以简化可维护性。
3. 。 。 。



# Hello World

```
// Java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

```
$ javac HelloWorld.java
$ java HelloWorld
Hello world!
```



# Hello World

```
// C++  
#include <iostream>  
using namespace std;  
  
int main() {  
    cout << "Hello world!" << endl;  
    return 0;  
}
```

```
$ g++ helloWorld.cpp  
$ ./a.out  
Hello world!
```



# Hello World

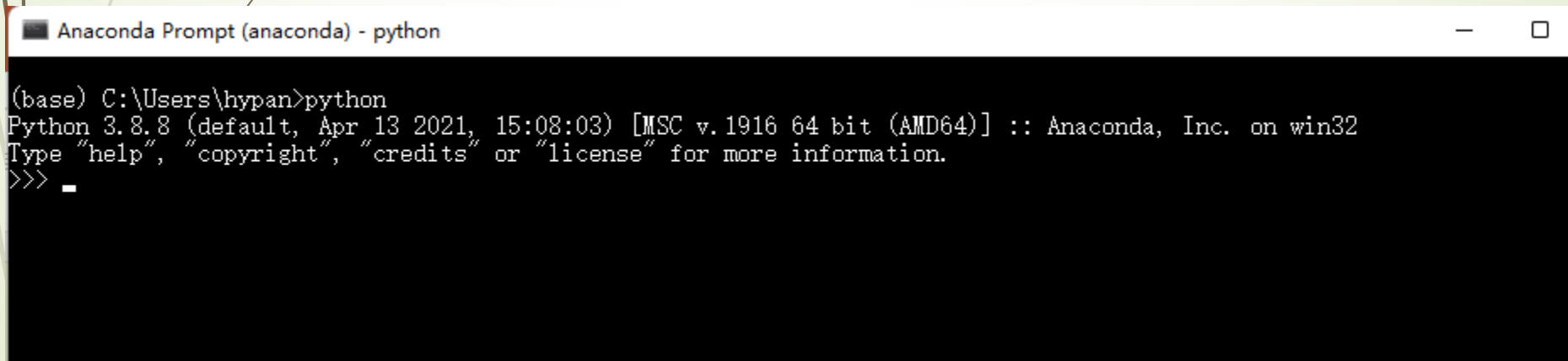
```
print("Hello world!")
```

```
$ python helloworld.py  
Hello world!
```

- Simple is better than complex.

# Python简单使用

- 下载和安装Python: Ubuntu/Windows
- 编辑器和IDE: **IDLE, PyCharm**, SublimeText, vim, emacs, Notepad++, Eclipse, etc.



```
Anaconda Prompt (anaconda) - python

(base) C:\Users\hypan>python
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> _
```

# Python简单使用

- 交互模式: **read-eval-print loop (REPL)**
  - it **Reads** input,
  - **Evaluates** it,
  - **Prints** it,
  - and **Loops** back to the beginning.

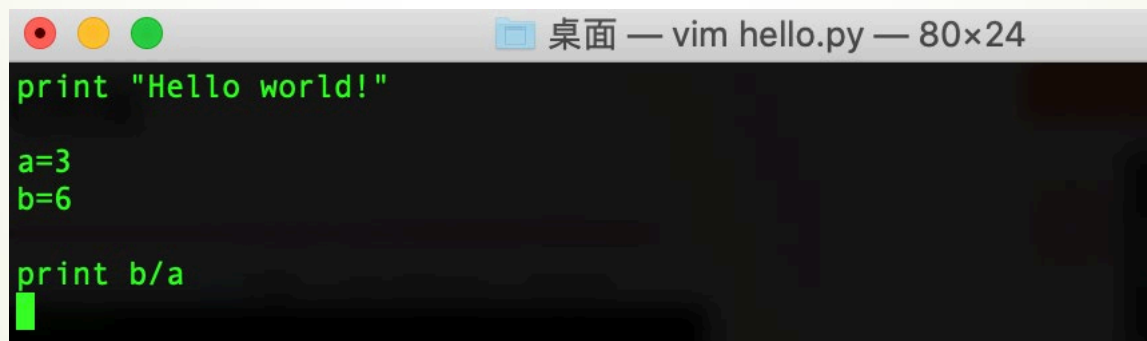
```
Python 2.7.16 (default, Nov  9 2019, 05:55:08)
[GCC 4.2.1 Compatible Apple LLVM 11.0.0 (clang-1100.0.32.4) (-macos10.15-objc-s
on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> print "Hello World!"
Hello World!
[>>> 4*5
20
[>>> a=3
[>>> b=6
[>>> b/a
2
[>>> █
```



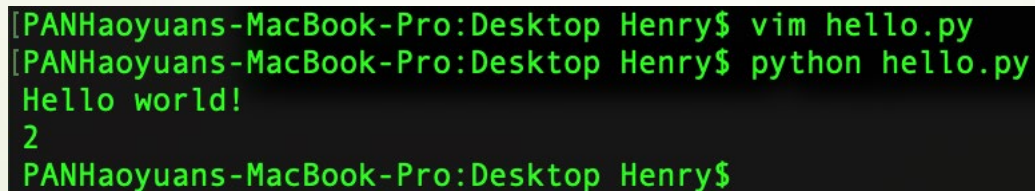
# Python简单使用

文件模式：

- 在命令行上执行Python程序
- 使用你喜欢的编辑器，比如vim，输入语句
- 然后执行，python + 文件名

A screenshot of a vim editor window. The title bar shows a folder icon, the text '桌面 — vim hello.py — 80x24', and three colored window control buttons (red, yellow, green). The editor area has a dark background with green text. The code is: 

```
print "Hello world!"  
  
a=3  
b=6  
  
print b/a
```

 A green cursor is at the end of the last line.A screenshot of a terminal window with a dark background and green text. It shows the following commands and output: 

```
[PANHaoyuans-MacBook-Pro:Desktop Henry$ vim hello.py  
[PANHaoyuans-MacBook-Pro:Desktop Henry$ python hello.py  
Hello world!  
2  
PANHaoyuans-MacBook-Pro:Desktop Henry$
```

# 编辑器和IDE

- 支持语法高亮显示，自动填充/完成功能
- 有一些支持集成调试和评测。

```
1 from gnuradio import gr
2 from gnuradio import eng_notation
3 from gnuradio.eng_option import eng_option
4 from gnuradio import digital
5 from optparse import OptionParser
6
7 from uhd_interface import uhd_transmitter
8 import transmit_path
9
10 import time, sys, struct, random
11
12
13 class my_top_block(gr.top_block):
14     def __init__(self, options):
15         gr.top_block.__init__(self)
16         if options.freq is not None:
17             self.d_tx_enable = True
18             u = uhd_transmitter(options.tx_args,
19                               options.bandwidth,
20                               options.tx_freq, options.tx_gain,
21                               options.spec, options.antenna,
22                               options.verbose, options.external)
23         elif options.outfile is not None:
24             self.d_tx_enable = False
25             u = gr.file_sink(gr.sizeof_gr_complex, options.outfile)]
26         else:
27             raise SystemExit("--freq or --outfile must be specified\n")
28
29         tx = transmit_path.ofdm_transmit_path(options)
30         self.connect(tx, u)
31
32         self.tx = tx
33         self.u = u
34
35     def send_pkt(self, payload='', time=None, eof=False, msgq_tx_enable=False):
36         if eof and not msgq_tx_enable:
37             self.tx.send_eof()
38         elif eof and msgq_tx_enable:
39             self.tx.send_eof2()
40
41         if self.d_tx_enable:
42             if msgq_tx_enable:
43                 self.tx.send_samples(payload, time)
44             else:
45                 self.tx.send_pkt(payload, time)
46
```

# Python的对象模型

*In Python, everything is an object and every object has a type.*

- 对象（object）是python语言中最基本的概念，  
在python中处理的一切都是对象。
- python中有许多**内置对象**可供编程者使用，内置对象可直接使用，如：数字、字符串、列表等
- **非内置对象**需要导入**模块**才能使用，如正弦函数 $\sin(x)$ ，随机数产生函数`random()`等。

```
import math
```

# 使用pip管理第三方包

## ■ pip工具常用命令

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 [&gt; 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

# 使用pip管理第三方包

- pip工具常用命令

```
C:\Users\panhy>pip

Usage:
  pip <command> [options]

Commands:
  install          Install packages.
  download         Download packages.
  uninstall        Uninstall packages.
  freeze           Output installed packages in requirements format.
  list             List installed packages.
  show             Show information about installed packages.
  check            Verify installed packages have compatible dependencies.
  config           Manage local and global configuration.
  search           Search PyPI for packages.
  wheel            Build wheels from your requirements.
  hash             Compute hashes of package archives.
  completion       A helper command used for command completion.
  help             Show help for commands.
```

# Python的对象模型

## 常用内置对象

对象类型	类型名称	示例	简要说明
数字	int, float, complex	1234, 1.0, 1.3e6, 1+2j	数字大小没有限制，内置支持复数及其运算
字符串	str	'shenzhen', "I'm student", "Python "	使用单引号、双引号、三引号作为定界符
列表	list	[1, 2, 3], ['a', 'b', 'c', 2]	所有元素放在一对方括号中，元素之间使用逗号分隔，其中的元素可以是任意类型
字典	dict	{1:'Alice', 2:'Bob', 3:'Carrie'}	所有元素放在一对大括号中，元素之间使用逗号分隔，元素形式为“键:值” (key: value)
元组	tuple	(1, 2, 3), (3,)	所有元素放在一对圆括号中，元素之间使用逗号分隔，如果元组中只有一个元素的话，后面的逗号不能省略

# Python的对象模型

## 常用内置对象

对象类型	类型名称	示例	简要说明
集合	set frozenset	{'a', 'b', 'c'}	所有元素放在一对大括号中，元素之间使用逗号分隔，元素不允许重复；另外，set是可变的，而frozenset是不可变的
布尔型	bool	True, False	逻辑值，关系运算符、成员测试运算符、同一性测试运算符组成的表达式的值一般为True或False
空类型	NoneType	None	空值
异常	Exception、 ValueError、 TypeError		Python内置大量异常类，分别对应不同类型的异常
文件		f = open('data.dat', 'rb')	open是Python内置函数，使用指定的模式打开文件，返回文件对象
其他迭代对象		生成器对象、range对象、zip对象、enumerate对象、map对象、filter对象等等	具有惰性求值的特点
编程单元		函数（使用def定义）、类（使用class定义）、模块（类型为module）	类和函数都属于可调用对象，模块用来集中存放函数、类、常量或其他对象

# Python变量

- 在Python中，不需要事先声明变量名及其类型，直接赋值即可创建各种类型的对象变量。Python解释器会根据赋值或运算来自动推断变量类型。

```
int x = 5;  
# C/C++/Java
```

```
x = 5  
# Python - 不需要写变量类型，不需要分号!
```

- Python还是一种动态类型语言（dynamic type），变量的类型也是可以随时变化的。

```
>>> x = 1  
>>> type(x) <class 'int'>
```

```
>>> x = "Shenzhen"  
>>> type(x) <class 'str'>
```



# 数字

**int:** 整数

**float:** 浮点数, 小数, 15.0、0.37、-11.2、1.2e2、314.15e-2

**complex:** 复数

5 # => 5 (int)

5.0 # => 5.0 (float)

1 + 8 # => 9 (int)

8 + 5.1 => 13.1 (float)

2 \* 4 # # => 8 (int)

2 / 3 # => 0.666666 (float)

7 // 3 # => 2 (int; 整数相除 integer division)

7 % 3 # => 1 (int; 取余 modulo operator)

3 \*\* 3 # => 27 (int; 幂运算exponential operator)

# 数字

- ❖ Python中的整数类型可以分为：
  - 十进制整数如，0、-1、9、123
  - 十六进制整数，需要16个数字0、1、2、3、4、5、6、7、8、9、a、b、c、d、e、f来表示整数，必须以0x开头，如0x10、0xfa、0xabcdef
  - 八进制整数，只需要8个数字0、1、2、3、4、5、6、7来表示整数，必须以0o开头，如0o35、0o11
  - 二进制整数，只需要2个数字0、1来表示整数，必须以0b开头如，0b101、0b100

# 数字

- Python内置支持复数类型

```
>>> a = 3+4j
>>> b = 5+6j
>>> c = a+b
```

```
>>> c
(8+10j)
```

```
>>> c.real
8.0
```

#查看复数实部

```
>>> c.imag
10.0
```

#查看复数虚部

```
>>> a.conjugate()
(3-4j)
```

#返回共轭复数

```
>>> a*b
(-9+38j)
```

#复数乘法

```
>>> a/b
```

#复数除法

```
(0.6393442622950819+0.03278688524590165j)
```

为什么这里一会有括号，  
一会又没有括号??

# 运算符和表达式

## Python运算符与功能

运算符	功能说明
+	算术加法，列表、元组、字符串合并与连接，正号
-	算术减法，集合差集，相反数
*	算术乘法，序列重复
/	真除法
//	求整商，但如果操作数中有实数的话，结果为实数形式的整数
%	求余数，字符串格式化
**	幂运算
<、<=、>、>=、==、!=	（值）大小比较，集合的包含关系比较
or	逻辑或
and	逻辑与
not	逻辑非
in	成员测试
is	对象同一性测试，即测试是否为同一个对象或内存地址是否相同
、^、&、<<、>>、~	位或、位异或、位与、左移位、右移位、位求反
&、 、^	集合交集、并集、对称差集
@	矩阵相乘运算符

# 算数运算符

- ❖ 赋值运算符或复合赋值运算符（例如+=、\*=）：表示创建变量或修改变量的值。
- ❖ 适用于使用下标来访问列表、字典等可变序列以及其他自定义对象中元素的情况。

`x += 5 # => x = x + 5`

`x -= 3 # => x = x - 3`

`x *= 2 # => x = x * 2`

`x /= 2 # => x = x / 2`

`x //= 4 # => x = x // 4`

`x %= 3 # => x = x % 3`

`x **= 3 # => x = x ** 3`

# 类型转换和舍入

- **Python2.x** 必须将5.0转换为5并执行int乘法，或者将2转换为2.0执行float乘法。
- 通过添加“.0”，可以将int转换为float
- float转换为int将丢失信息
- 在混合类型表达式中，Python将把int转换为float

```
>>> 7/3      #Python 2.x
2
>>> 7.0/3
2.3333333333333335
>>> 7.0//3
2.0
>>> 7//3
2
```

```
>>> float(22//5)
4.0
>>> int(4.5)
4
>>> int(3.9)
3
>>> round(3.9)
4
>>> round(3.1415926, 2)
3.14
```

# Python内存管理方式

- 在Python中，允许多个变量指向同一个值，例如：

```
>>> x = 3
```

```
>>> id(x)
```

```
1786684560
```

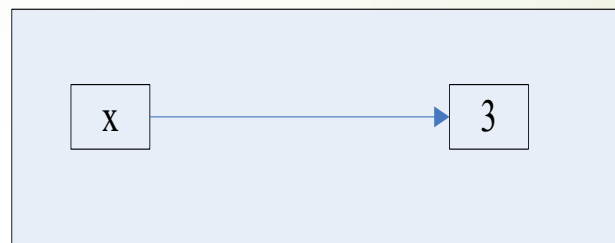
← 获取对象的内存地址

```
>>> y = x
```

```
>>> id(y)
```

```
1786684560
```

← 一样的对象



- 然而，当为其中一个变量修改值以后，其内存地址将会变化，但这并不影响另一个变量，例如接着上面的代码再继续执行下面的代码：

```
>>> x += 6
```

```
>>> id(x)
```

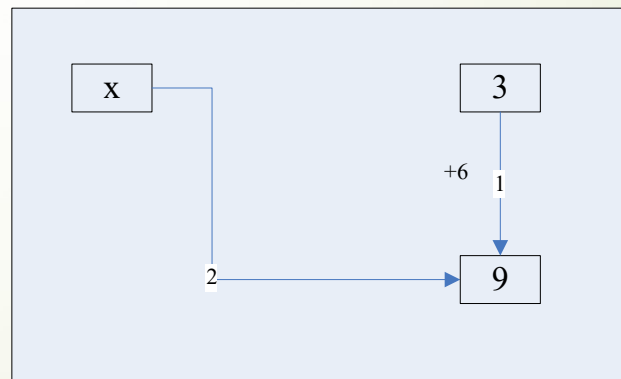
```
1786684752
```

```
>>> y
```

```
3
```

```
>>> id(y)
```

```
1786684560
```



# Python内存管理方式

- 结论：Python采用的是基于值的内存管理方式，如果为不同变量赋值为相同值，这个值在内存中只有一份，多个变量指向同一块内存地址。

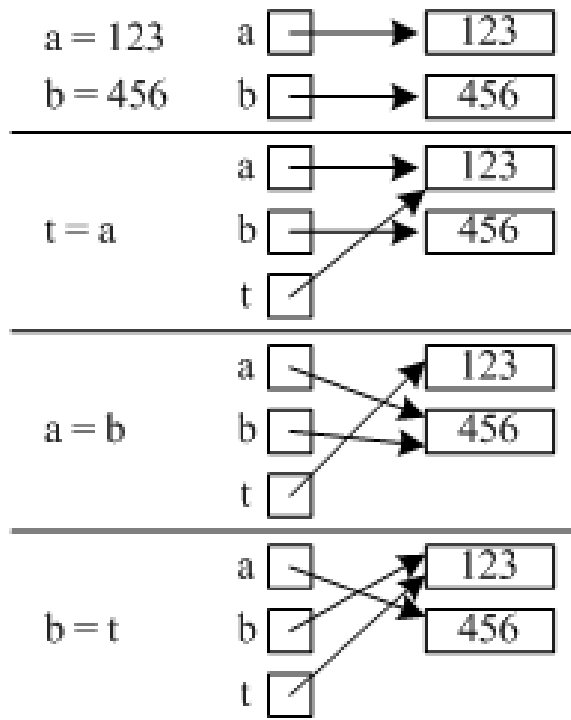
```
Python 2.7.16 (default, Dec 13 2019, 18:00:32)
[GCC 4.2.1 Compatible Apple LLVM 11.0.0 (clang-1100.0.32.4) (-macos10.15-
on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> x=3
>>> id(x)
140435662077336
>>> y=3
>>> id(y)
140435662077336
>>> □
```



# Python内存管理方式

- 更多例子

```
>>> a=123 #a指向值为123的int型实例对象
>>> b=456 #b指向值为456的int型实例对象
>>> t=a    #变量t和a一样，指向（引用）对象实例123
>>> a=b    #变量a和b一样，指向（引用）对象实例456
>>> b=t    #变量b和t一样，指向（引用）对象实例123
```



# 布尔运算

■ 布尔值是整型（int）的子类，用于逻辑判断真（True）或假（False），用数值1和0分别代表常量True和False。

■ 比较运算符

```
not True # => False
True and False # => False
True or False # => True
```

```
5 == 5 # => True
1 != 100 # => True
3 > 6 # => False
4 >= 9 # => False
3 > 2 > 1 # => True (3 > 2 and 2 > 1)
```

# 布尔运算

or

P	Q	P or Q
T	T	T
T	F	T
F	T	T
F	F	F

and

P	Q	P and Q
T	T	T
T	F	F
F	T	F
F	F	F

not

P	not P
T	F
F	T

# 布尔运算

- ▶ `a or not b and c`
- ▶ 优先级, 从高到低, `not`, `and`, `or`.
- ▶ `(a or ((not b) and c))`
- ▶ 如果不记住布尔优先规则, 用括号来防止混淆
- ▶ `a or true # true`
- ▶ `not(not a) # a`
- ▶ `a or (b and c) # (a or b) and (a or c)`  
`a and (b or c) # (a and b) or (a and c)`
- ▶ `not(a or b) # (not a) and (not b)`  
`not(a and b) # (not a) or (not b)`

# bool数据类型的转换

>>> bool()	#空置转布尔类型
False	
>>> bool(0)	#整数0转布尔值
False	
>>> bool(1)	#整数1转布尔值
True	
>>> bool(100)	#整数100转布尔值
True	
>>> bool([])	#空列表转布尔值
False	
>>> bool([1, 2])	#非列表转布尔值
True	
>>> bool('a')	#字符串转布尔值
True	

# 字符串 String

- 由单引号或者双引号括起来的字符串。
- 自带的字符串操作函数 (<https://docs.python.org/>).
- 空串表示为'' 或 ""

```
>>> bool('')
```

#空串转布尔值?

- 字符串的索引

	0	1	2	3	4	5	6	7	8	9	10	11	
string =	"	h	a	p	p	y	,		c	o	d	e	"

# 字符串 String

	0	1	2	3	4	5	6	7	8	9	10	11		
string =	"	h	a	p	p	y	,			c	o	d	e	"

```
>>> course = "happy, code"
>>> course[2] 'p' # Index at individual elements
>>> course[:3] 'hap' # Index from beginning to element (exclusive)
>>> course[5:] ', code' # Index from element to end (inclusive)
```

切片功能（非常重要非常好用！）

# 列表 List

- Python 的数组或者Vector
- 最常用的Python数据类型，它可以作为一个方括号内的逗号分隔值出现。
- 列表的数据项不需要具有相同的类型
- 创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来。
- 与字符串的索引一样，列表索引从0开始。



# 列表 List

*#初始化新的list*

```
empty = []  
letters = ["a", "b", "c"]  
numbers = [1, 2, 3]
```

*#数据项不需要具有相同的类型，可以是list*

```
many_types = ["a", 2, 3, [4, 5]]
```

*# 小问1- many\_types[3] 是什么?*

*# [4, 5]*

*# 小问2- many\_types[3][1]是什么?*

*# 5*

*# list 的自带函数*

```
>>> numbers.extend([4, 5, 6])  
>>> numbers  
[1, 2, 3, 4, 5, 6]
```

# 列表 List: 索引

*# 初始化新的list*

```
empty = []  
letters = ["a", "b", "c"]  
numbers = [1, 2, 3, 4]
```

*# 索引规则与字符串一样*

```
>>> letters[1]  
'b'  
>>> numbers[1:-1]  
[2, 3]  
>>> numbers[::2]  
[1, 3]
```

# 列表 List: 自带函数

# 长度

```
>>> len([])
```

```
0
```

```
>>> len("China")
```

```
5
```

```
>>> len([1, 2, 3, "Parth"])
```

```
4
```

# 检查元素是否在list里面

```
>>> 0 in [2, 3, 4]
```

```
False
```

```
>>> "5" in [3, 4, 5, "CS"]
```

```
False # ???
```

# 检查子字符串

```
>>> "tho" in "Python"
```

```
True
```



# 更多的数据结构

- Tuple
- Dictionary
- Set
- ...

# 选择与循环

- 在选择和循环结构中，**条件表达式**的值只要不是False、0（或0.0、0j等）、空值None、空列表、空元组、空集合、空字典、空字符串、空range对象或其他空迭代对象，Python解释器均认为与True等价。
- 从这个意义上来讲，几乎所有的Python合法表达式都可以作为条件表达式，包括含有函数调用的表达式。

```
>>> if 3:
    print(5)
5

>>> a = [1, 2, 3]
>>> if a:
    print(a)
[1, 2, 3]

>>> a = []
>>> if a:
    print(a)
else:
    print('empty')
empty
```

# 选择

## ➤ if 语句

```
if expression:  
    statements
```

## ➤ 每个条件后面要使用冒号“:”，表示满足条件后需要执行的语句块，后面几种其它形式的选择结构和循环结构中是冒号也是必须要有的

## ➤ 如果boolean语句为真 (=1)，执行statements；否则 (=0)，跳过statements。

```
a = 1  
b = 0  
if a:  
    print("a is true!")  
if not b:  
    print("b is false!")  
if a and b:  
    print("a and b are true!")  
if a or b:  
    print("a or b is true!")
```

输出:

```
a is true!  
b is false!  
a or b is true!
```

# 选择

## ➡ if -- else

**if** expression:  
    statements

**elif** expression:  
    statements

**else:**  
    statements

```
a = 1
b = 0
c = 2
if a > b:
    if a > c:
        print("a is greatest")
    else:
        print("c is greatest")
elif b > c:
    print("b is greatest")
else:
    print("c is greatest")
```

输出:  
c is greatest

# 选择

- 在Python中，条件表达式中不允许使用赋值运算符“=”，避免了其他语言中误将关系运算符“==”写作赋值运算符“=”带来的麻烦，例如下面的代码，在条件表达式中使用赋值运算符“=”将抛出异常，提示语法错误。

```
>>> if a=3:  
SyntaxError: invalid syntax
```

```
>>> if (a=3) and (b=4):  
SyntaxError: invalid syntax
```



# 选择

- Python还支持如下形式的表达式:

`value1 if condition else value2`

当条件表达式`condition`的值与`True`等价时, 表达式的值为`value1`, 否则表达式的值为`value2`。另外, 在`value1`和`value2`中还可以使用复杂表达式, 包括函数调用和基本输出语句。

```
>>> a = 5
```

```
>>> print(6) if a>3 else print(5)
```

```
6
```

```
>>> print(6 if a>3 else 5)
```

```
6
```

```
>>> b = 6 if a>13 else 9
```

```
>>> b
```

```
9
```

# 选择结构的嵌套

```
if 表达式1:  
    语句块1  
    if 表达式2:  
        语句块2  
    else:  
        语句块3  
else:  
    if 表达式4:  
        语句块4
```

```
1 | if 表达式 1:  
  |   语句块 1  
  |   if 表达式 2:  
2 |     3 | 语句块 2  
  |   else:  
  |     3 | 语句块 3  
  | else:  
  |   2 | if 表达式 4:  
    |   3 | 语句块 4
```

注意：缩进必须要正确并且一致！！！！

缩进必须要正确并且一致！！！！

缩进必须要正确并且一致！！！！

# 循环

## ➤ for 循环

**for** var **in** sequence:  
statements

➤ for执行时，依次将可迭代对象sequence中的值赋给变量var，变量var每赋值一次，则执行一次循环体statements

➤ sequence可以是string, lists, 和其他的数据结构

```
>>> for ch in "this":  
        print(ch)
```

```
t  
h  
i  
s
```

```
>>> for num in [3, 1, 4, 1, 5]:  
        print(num**2)
```

```
9  
1  
16  
1  
25  
>>>
```

# 循环

- ▶ range(), for语句的好伙伴
- ▶ 语法: range(stop) or range(start, stop[, step])

```
>>> range(3) # 生成 0, 1, 2
```

```
>>> range(5, 10) # 生成 5, 6, 7, 8, 9
```

```
>>> range(2, 12, 3) # 生成 2, 5, 8, 11
```

```
>>> range(-7, -30, -5) # 生成 -7, -12, -17, -22, -27
```

```
>>> for i in range(0,8,2): # 输出 0, 2, 4, 6
    print i
```

# 循环

- 控制for循环: **break**, **continue**, **pass**.
- **break**语句在while循环和for循环中都可以使用,一般放在if选择结构中,一旦**break**语句被执行,将使得整个循环提前结束。
- **continue**语句的作用是终止当前循环,并忽略**continue**之后的语句,然后回到循环的顶端,提前进入下一次循环。

```
for num in range(10,20):  
    if num%2 == 0:  
        continue  
    for i in range(3,num):  
        if num%i == 0:  
            break  
else:  
    print(num, " is a prime number")
```

输出:

```
11 is a prime number  
13 is a prime number  
17 is a prime number  
19 is a prime number
```

# 循环

## ➤ While 循环

- 当不知道循环次数，但知道循环条件时，一般使用while语句。
- 与for循环类似，可在循环体中使用break和continue语句。

```
while expression:  
    statements
```

```
i = 1  
while i < 4:  
    print(i)  
    i = i + 1
```

```
flag = True
```

```
while flag and i < 8:  
    print(flag, i)  
    i = i + 1
```

输出:

1

2

3

True 4

True 5

True 6

True 7

# 基本输入输出

- 用Python进行程序设计，输入是通过input( )函数来实现的，input( )的一般格式为：

```
x = input(" ")
```

- 该函数返回输入的对象。
- 可输入数字、字符串和其它任意类型对象。
- Python 2.x和Python 3.x稍有不同

# 基本输入输出

- 在Python 2.x中，该函数返回结果的类型由输入值时所使用的界定符来决定，

```
>>> x = input("Please input:")
Please input:3                #没有界定符，整数
>>> print type(x)
<type 'int'>
```

```
>>> x = input("Please input:")
Please input:'3'              #单引号，字符串
>>> print type(x)
<type 'str'>
```

```
>>> x = input("Please input:")
Please input:[1,2,3]          #方括号，列表
>>> print type(x)
<type 'list'>
```



# 基本输入输出

- 在Python 2.x中，还有另外一个内置函数 `raw_input()`。与 `input()` 函数不同的是，`raw_input()` 函数返回结果的类型一律为字符串，而不论用户使用什么界定符。

```
>>> x = raw_input("Please input:")
Please input:[1, 2, 3]
>>> print type(x)
<type 'str'>
```

# 基本输入输出

- 在Python 3.x中，不存在raw\_input()函数只提供了input()函数用来接收用户的键盘输入。
- 在Python 3.x中，不论用户输入数据时使用什么界定符，input()函数的返回结果都是字符串，需要将其转换为相应的类型再处理。

```
>>> x = input('Please input:')
```

```
Please input:3
```

```
>>> print(type(x))
```

```
<class 'str'>
```

```
>>> x = input('Please input:')
```

```
Please input:'1'
```

```
>>> print(type(x))
```

```
<class 'str'>
```

```
>>> x = input('Please input:')
```

```
Please input:[1,2,3]
```

```
>>> print(type(x))
```

```
<class 'str'>
```

```
>>> x = raw_input('Please input:')
```

```
NameError: name 'raw_input' is not defined
```

# 基本输入输出

❖ Python 2.x和Python 3.x的输出方法也不一样。

✓ 在Python 2.x中，使用print语句进行输出

✓ Python 3.x中使用print()函数进行输出。

■ 另外一个重要的不同是，对于Python 2.x而言，在print语句之后加上逗号“,”则表示输出内容之后不换行，例如：

```
>>> for i in range(10):
```

```
    print i,
```

```
0 1 2 3 4 5 6 7 8 9
```

■ 在Python 3.x中，为了实现上述功能则需要使用下面的方法：

```
>>> for i in range(10,20):
```

```
    print(i, end=' ')
```

```
10 11 12 13 14 15 16 17 18 19
```

# Python格式注意

## (1) 缩进

- ✓ 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。

```
# here's a comment
for i in range(0,3):
    print I

def myfunc():
    """here's a comment about
    the myfunc function"""
    print "I'm in a function!"
```

# Python格式注意

- ▶ “Whitespaces are ugly.”
  - ▶ You'll get used to it.

## Python

```
while x==1:  
    ....if y:  
    .....f1()  
    ....f2()
```

## C/C++/Java...

```
while (x==1) {  
    if (y) { f1();}  
    f2();  
}
```

# Python格式注意

## (2) 注释

常用的注释方式主要有两种：

- ✓ 以#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

```
# here's a comment
```

```
for i in range(0,3):  
    print I
```

```
def myfunc():
```

```
    """here's a comment about  
    the myfunc function"""
```

```
    print "I'm in a function!"
```

# 练习

- Fibonacci 序列：从第三项开始，每一项等于前两项的和
- 假设从1, 2开始，前面10项是 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...
- 练习1：输出第11项到第20项
- 练习2：输出前20项的偶数
- 练习3：计算Fibonacci 序列中不超过4000000的所有偶数的和

# 练习

```
#1
f1, f2 = 1, 2
for i in range(1,21):
    if i>10:
        print f1
    f1, f2 = f2, f1 + f2

print "\n"

#2
f1, f2 = 1, 2
for i in range(1,21):
    if f1%2==0:
        print f1
    f1, f2 = f2, f1 + f2

print "\n"

#3
f1, f2 = 1, 2
total = 0
while f1 < 4000000:
    if f1 % 2 == 0:
        total = total + f1
    f1, f2 = f2, f1 + f2
print total
```

输出:

144  
233  
377  
610  
987  
1597  
2584  
4181  
6765  
10946

2  
8  
34  
144  
610  
2584  
10946

4613732