Python编程简介

目录

- Python起源
- 为什么要学习Python？
- 怎样高效学习Python？

- Python的开发环境
  - Jupyter Notebook
- Python基本编程
  - 数据类型和变量
  - 运算符和表达式
  - 容器类型
  - 控制流语句
  - 函数
  - 文件操作

Python起源

Guido van Rossum

(1956 -)

1989年末，Guido van Rossum为了打发圣诞节的无聊，创造了Python语言。2005年12月，入职Google工作。2012年12月，加入Dropbox公司。

TOBIE Index

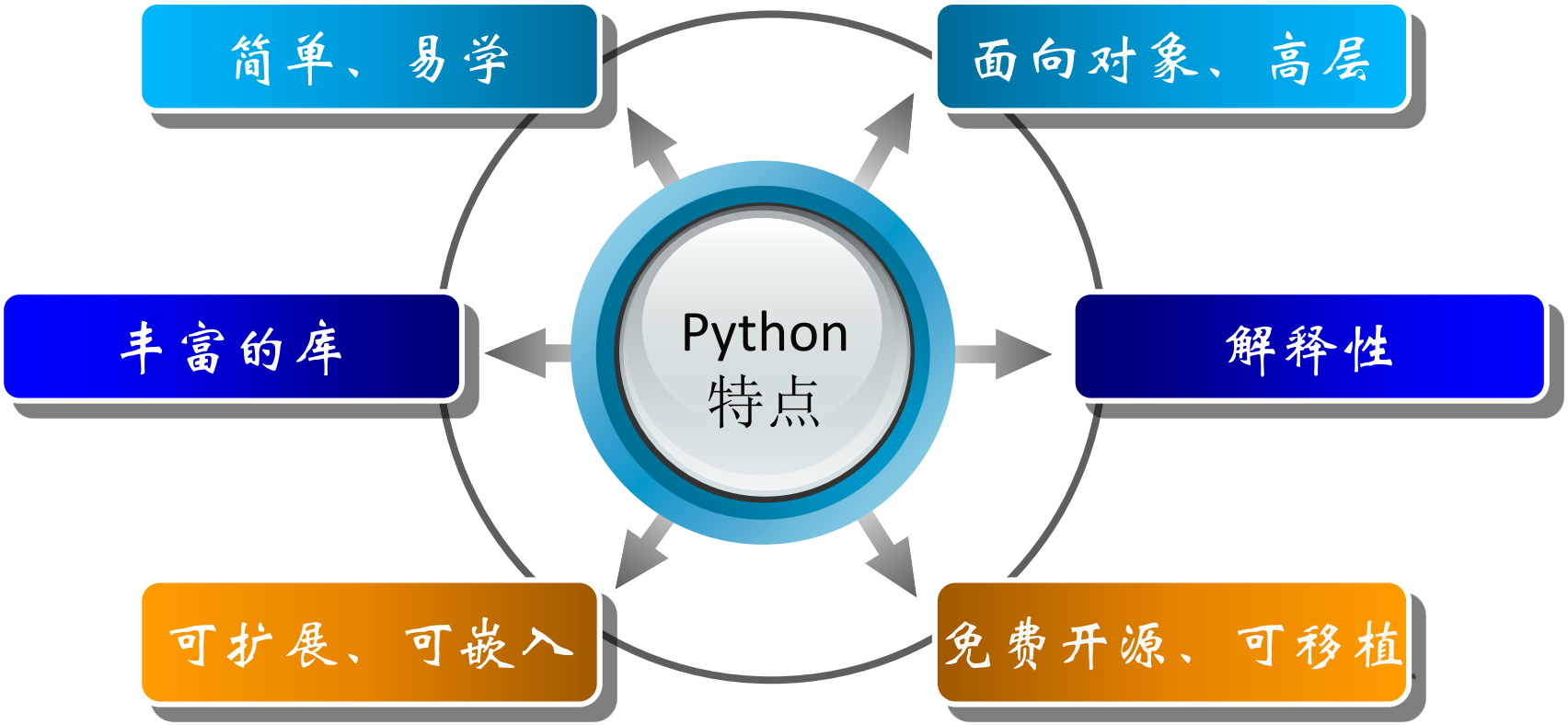
Sep 2019	Sep 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.661%	-0.78%
2	2		C	15.205%	-0.24%
3	3		Python	9.874%	+2.22%
4	4		C++	5.635%	-1.76%
5	6	^	C#	3.399%	+0.10%
6	5	v	Visual Basic .NET	3.291%	-2.02%
7	8	^	JavaScript	2.128%	-0.00%
8	9	^	SQL	1.944%	-0.12%
9	7	v	PHP	1.863%	-0.91%
10	10		Objective-C	1.840%	+0.33%
11	34	^^	Groovy	1.502%	+1.20%
12	14	^	Assembly language	1.378%	+0.15%
13	11	v	Delphi/Object Pascal	1.335%	+0.04%
14	16	^	Go	1.220%	+0.14%
15	12	v	Ruby	1.211%	-0.08%

IEEE top programming languages of 2019

Rank	Language	Type	Score
1	Python	web, desktop, system	100.0
2	Java	web, mobile, desktop	96.3
3	C	mobile, desktop, system	94.4
4	C++	mobile, desktop, system	87.5
5	R	desktop	81.5
6	JavaScript	web	79.4
7	C#	web, mobile, desktop, system	74.5
8	Matlab	desktop	70.6
9	Swift	mobile, desktop	69.1
10	Go	web, desktop	68.0

为什么要学Python？

- 人生苦短，我用Python！



Python的特点

- **简单、易学**
  - 一种代表简单主义思想的语言，有简单的语法，容易上手。
  - 伪代码本质是它最大的优点之一。
  - 使你能够专注于解决问题而不是去搞明白语言本身。
- **面向对象的高层语言**
  - 无需关注底层细节，而C/C++中需要操作指针。
  - 与其他语言相比，以强大而又简单的方式实现面向对象编程。
- **解释性**
  - 不需要被编译成二进制代码，可以直接在源代码上运行。
  - 对于编译性语言（C/C++），源文件->编译/链接器->可执行文件。

Python的特点

- **免费开源，可移植性**
  - Unix衍生系统，Win32系统家族，掌上平台（掌上电脑/手机），游戏控制台（PSP）等等。
- **可扩展性，可嵌入性**
  - 如果一段关键代码希望运行得更快或者希望算法不公开，你可以把这部分程序用C或C++编写，然后在Python程序中使用。
  - 可以把Python嵌入到C/C++程序中，从而向程序用户提供脚本功能。
- **丰富的库**
  - Python标准库确实很庞大，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器等等。此外还有其他高质量的库，如wxPython、Twisted和图像库等等。

如何高效学习Python?

- 不需要具有计算机专业的学位
- 不需要上一门完整的Python编程课
- 不需要记住所有的语法

- 自顶向下学习法：
  - 先学习核心的编程概念
  - 再学习使用一些相关的库
  - 最后通过实际项目来运用知识和改进技术

Python 2还是Python 3？

- Python 3不向下兼容Python 2，即Python 2中的一些函数和包不能在Python 3中使用
- 2010年发布的Python 2.7是Python 2最后的版本，Python 2将不再更新
- 2020年1月1日起，Python核心开发团队将不再对Python 2提供任何官方支持
- Python 3是未来的主流

- 本课程中使用Python 3

Python的开发环境

- Anaconda
  - 一个用于科学计算的Python发行版
  - 提供了包管理与环境管理的功能，可以很方便地解决多版本python并存、切换以及各种第三方包安装问题
  - 支持 Linux, Mac OS, Windows系统
  - 下载：https://www.anaconda.com/distribution/

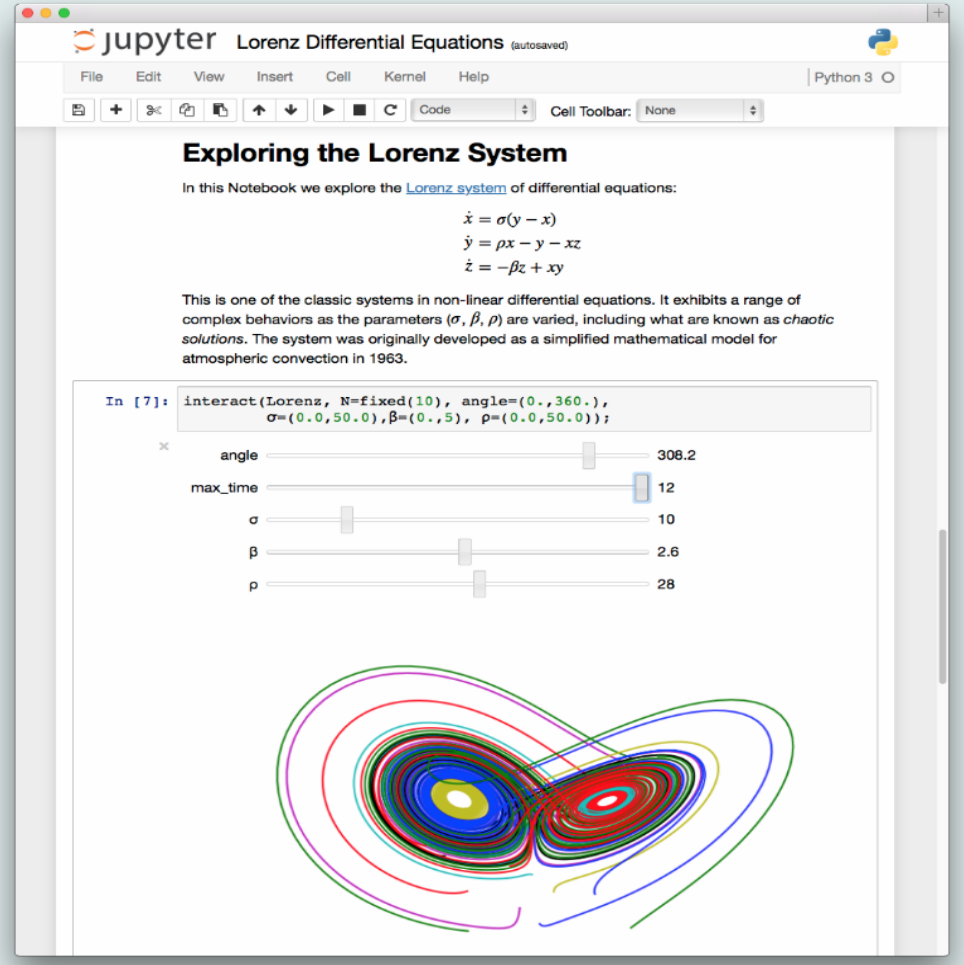
Anaconda里的其他包

- **NumPy**（ http://www.numpy.org/ ）：用于处理（大）数组
- **Pandas**（http://pandas.pydata.org/）：数据分析工具包
- **Matplotlib**（http://matplotlib.org）：用于绘制图表
- SciPy（http://www.scipy.org）：包含许多有用的科学函数
- **Scikit-learn**（http://scikit-learn.org/）：机器学习算法

- **IPython**（http://ipython.org/）：基于Shell或浏览器的开发环境
- Spyder（ https://www.spyder-ide.org/ ）：交互式集成开发环境

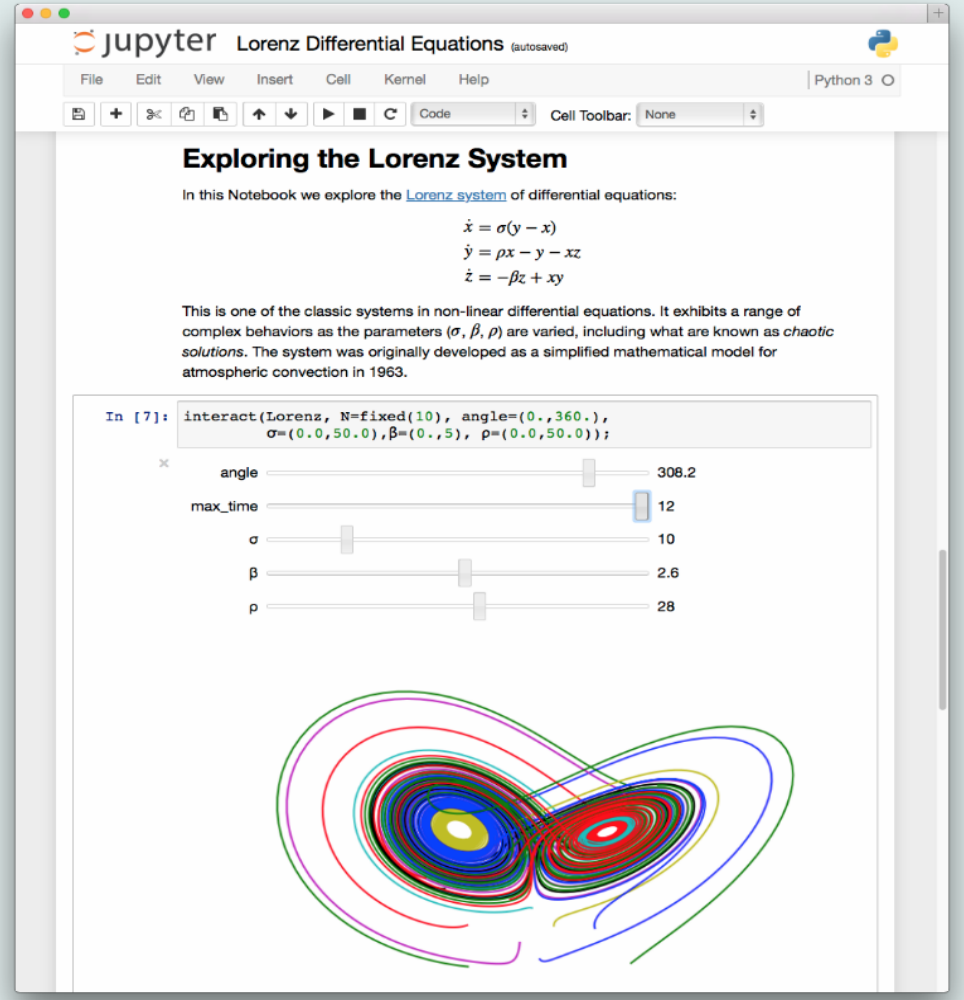
Jupyter Notebook

- 此前被称为IPython Notebook
- 多语分析环境—支持40多种编程语言
- Jupyter 是 Julia，Python和R几个词的变位词
- 支持多种内容类型：代码、描述文本、图像、视频等等
- http://jupyter.org/



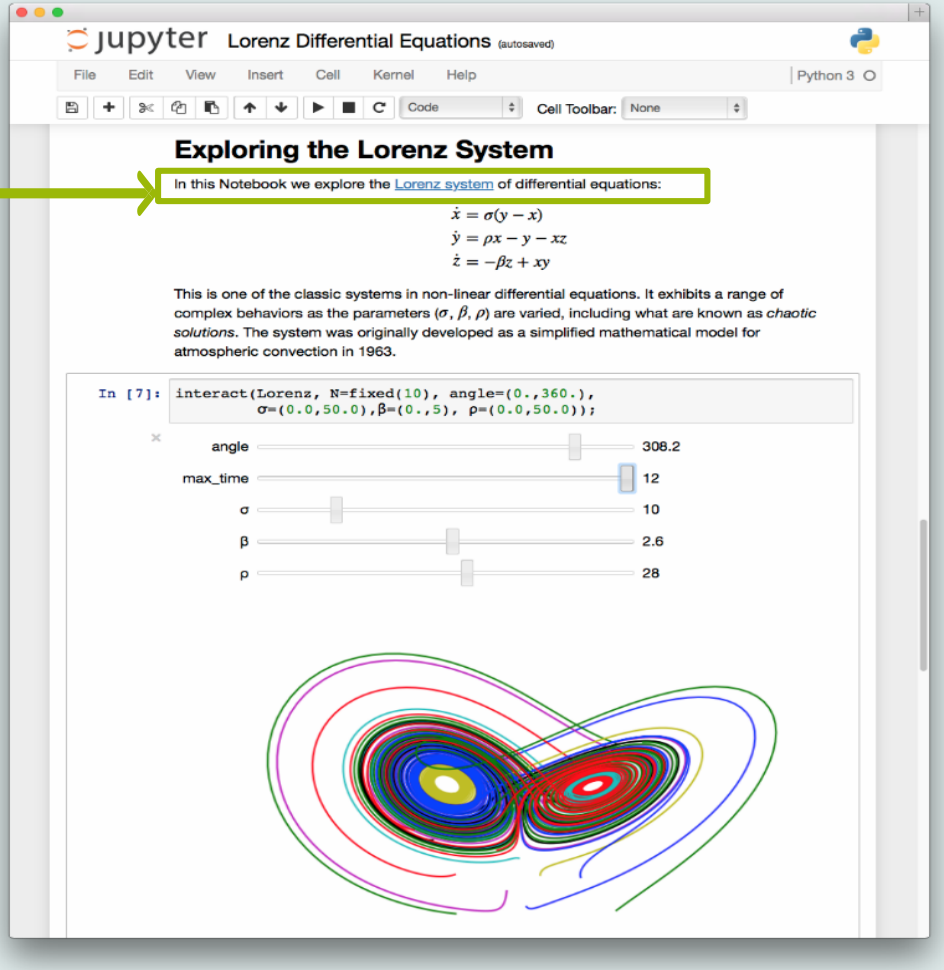
Jupyter Notebook

- HTML & Markdown
- LaTeX（公式）
- Code（代码）



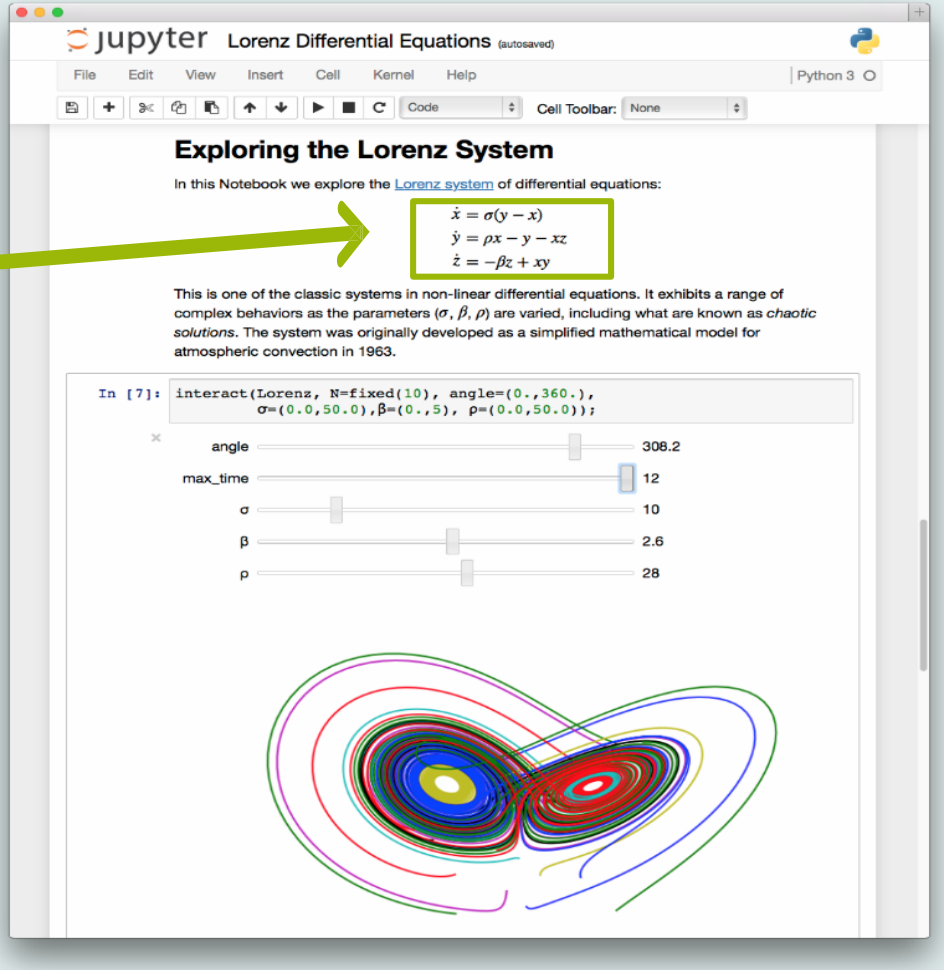
Jupyter Notebook

- HTML & Markdown
- LaTeX（公式）
- Code（代码）



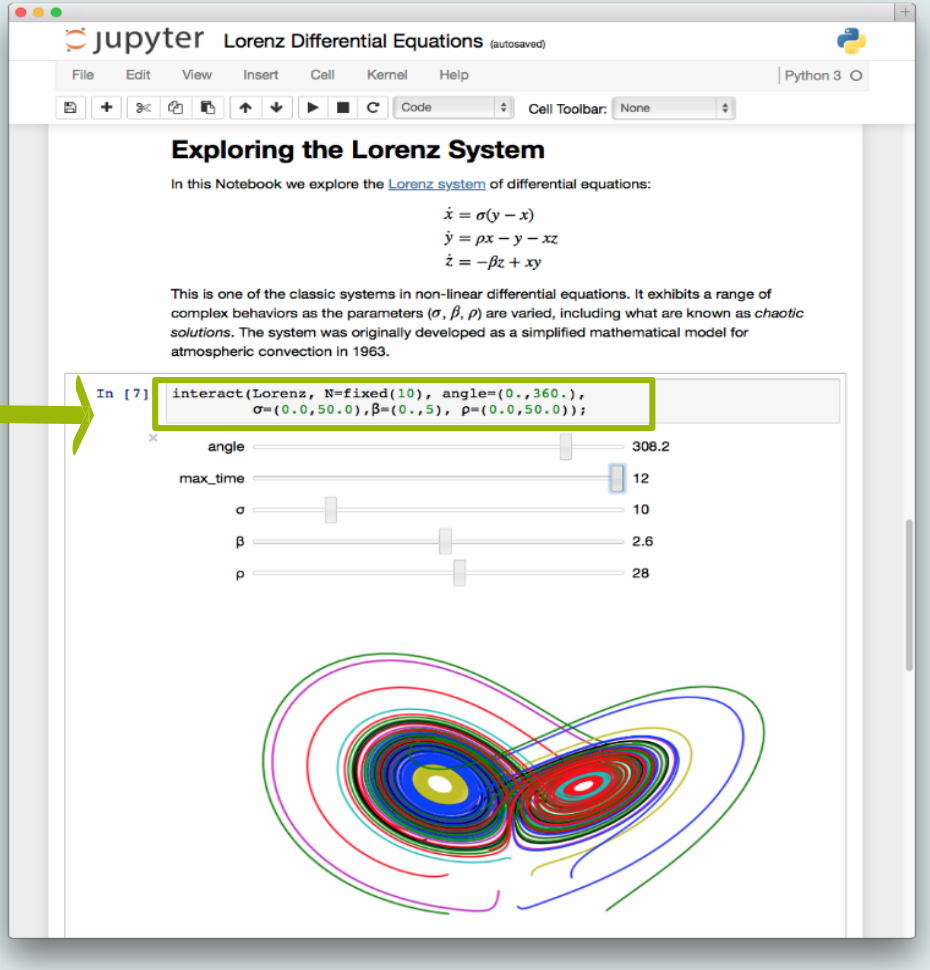
Jupyter Notebook

- HTML & Markdown
- LaTeX（公式）
- Code（代码）



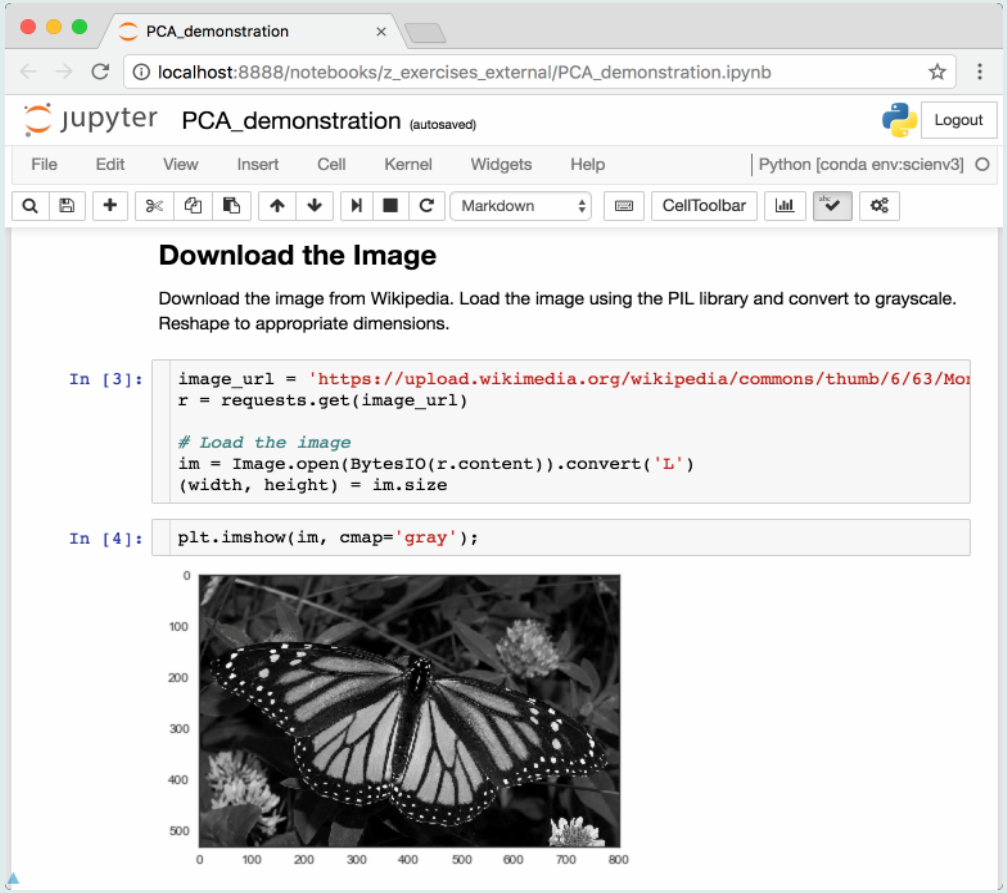
Jupyter Notebook

- HTML & Markdown
- LaTeX（公式）
- Code（代码）



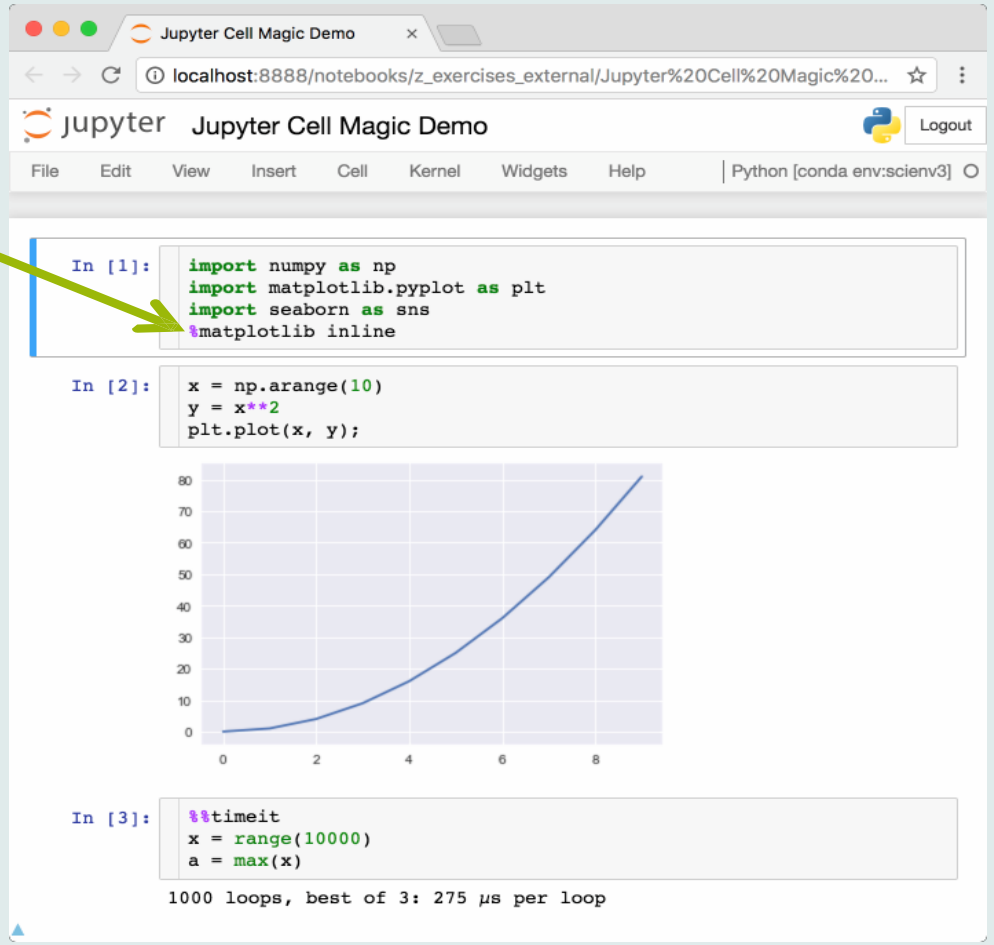
Jupyter Notebook

- 代码被划分成多个单元，可以控制执行过程
- 允许进行交互式开发
- 非常适合探索式分析与建模



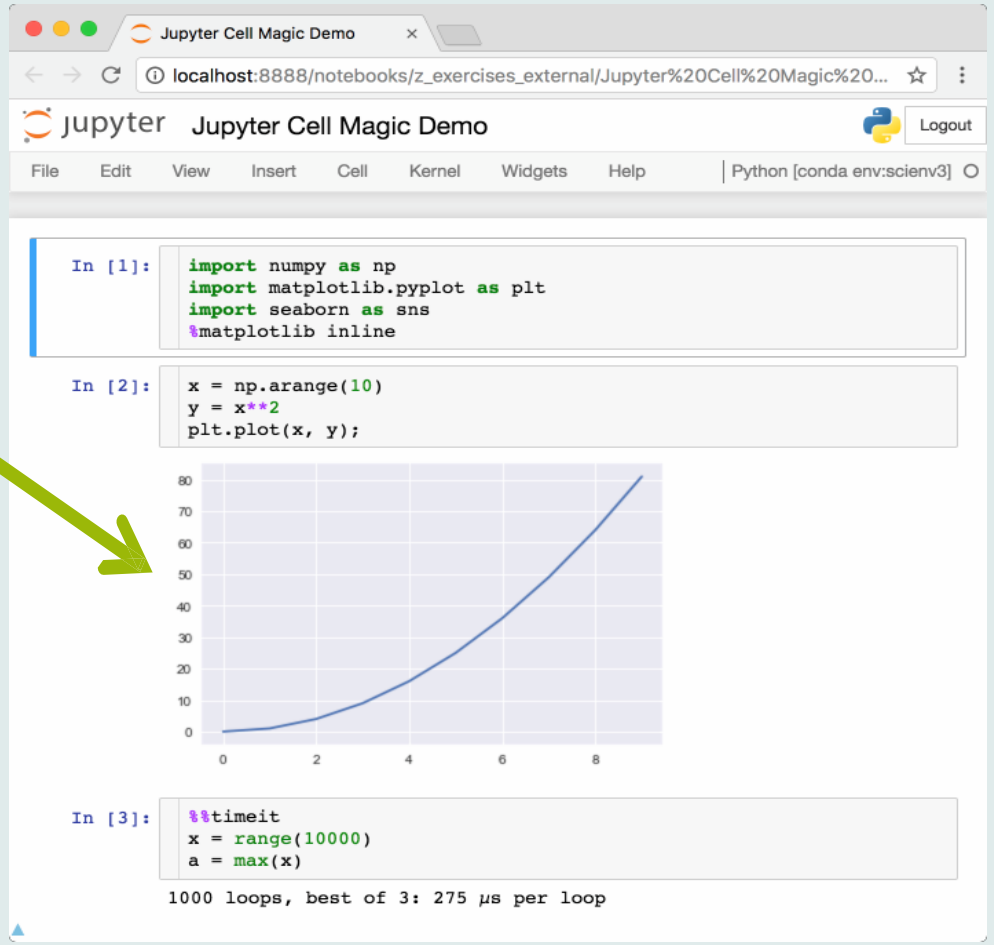
Jupyter魔术命令

- %matplotlib inline
  - 将图表显示在Jupyter notebook中



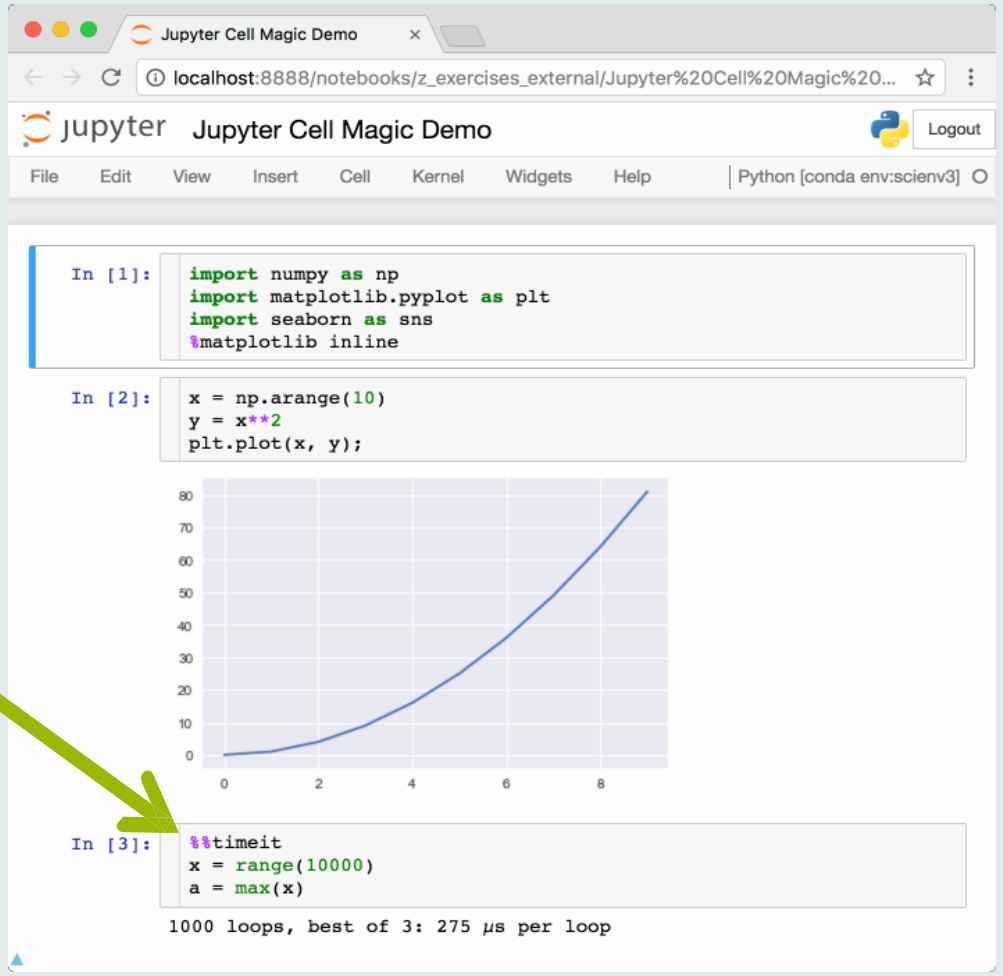
Jupyter魔术命令

- %matplotlib inline
  - 将图表显示在Jupyter notebook中



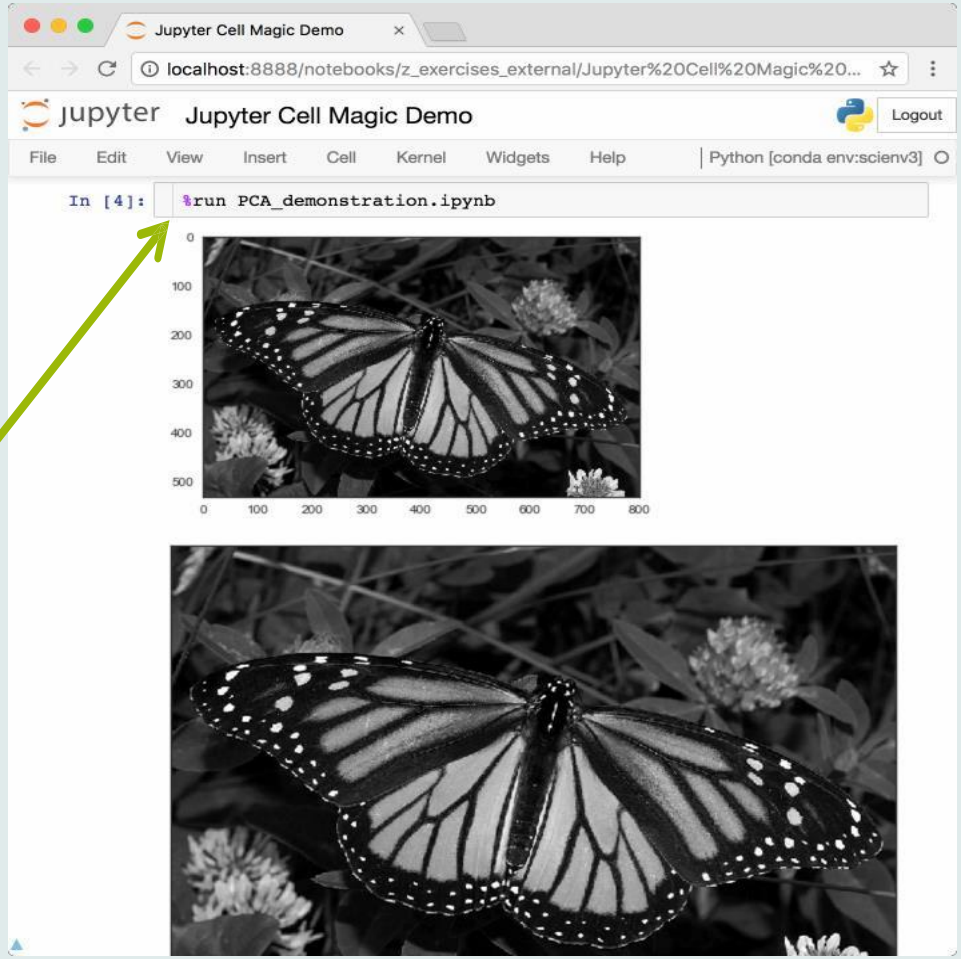
Jupyter魔术命令

- %matplotlib inline
  - 将图表显示在Jupyter notebook中
- %%timeit
  - 记录一个单元的执行时间



Jupyter魔术命令

- %matplotlib inline
  - 将图表显示在Jupyter notebook中
- %%timeit
  - 记录一个单元的执行时间
- %run filename.ipynb
  - 运行另一个notebook或python文件的代码

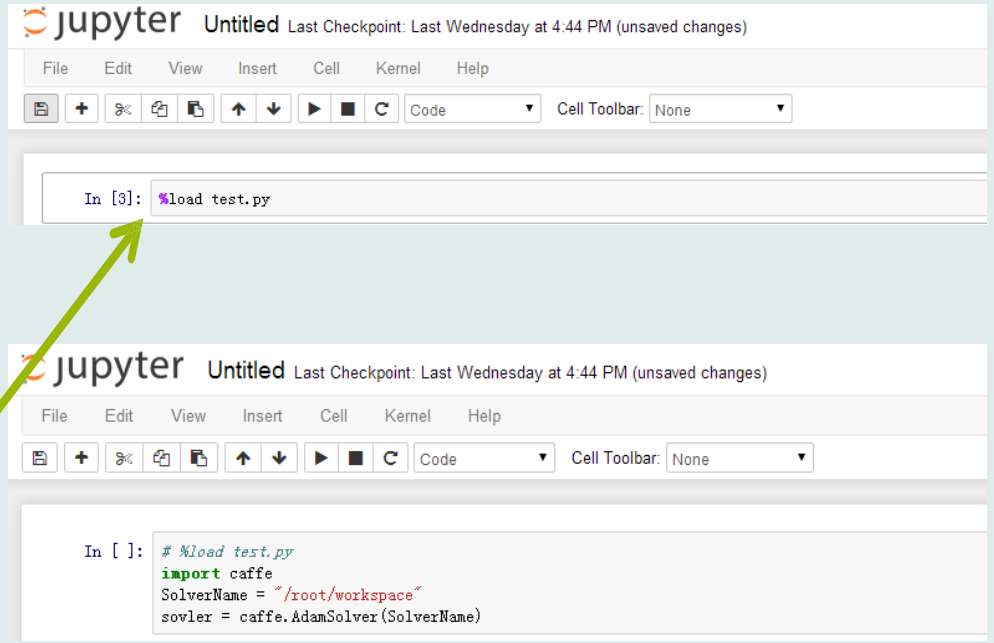


Jupyter魔术命令

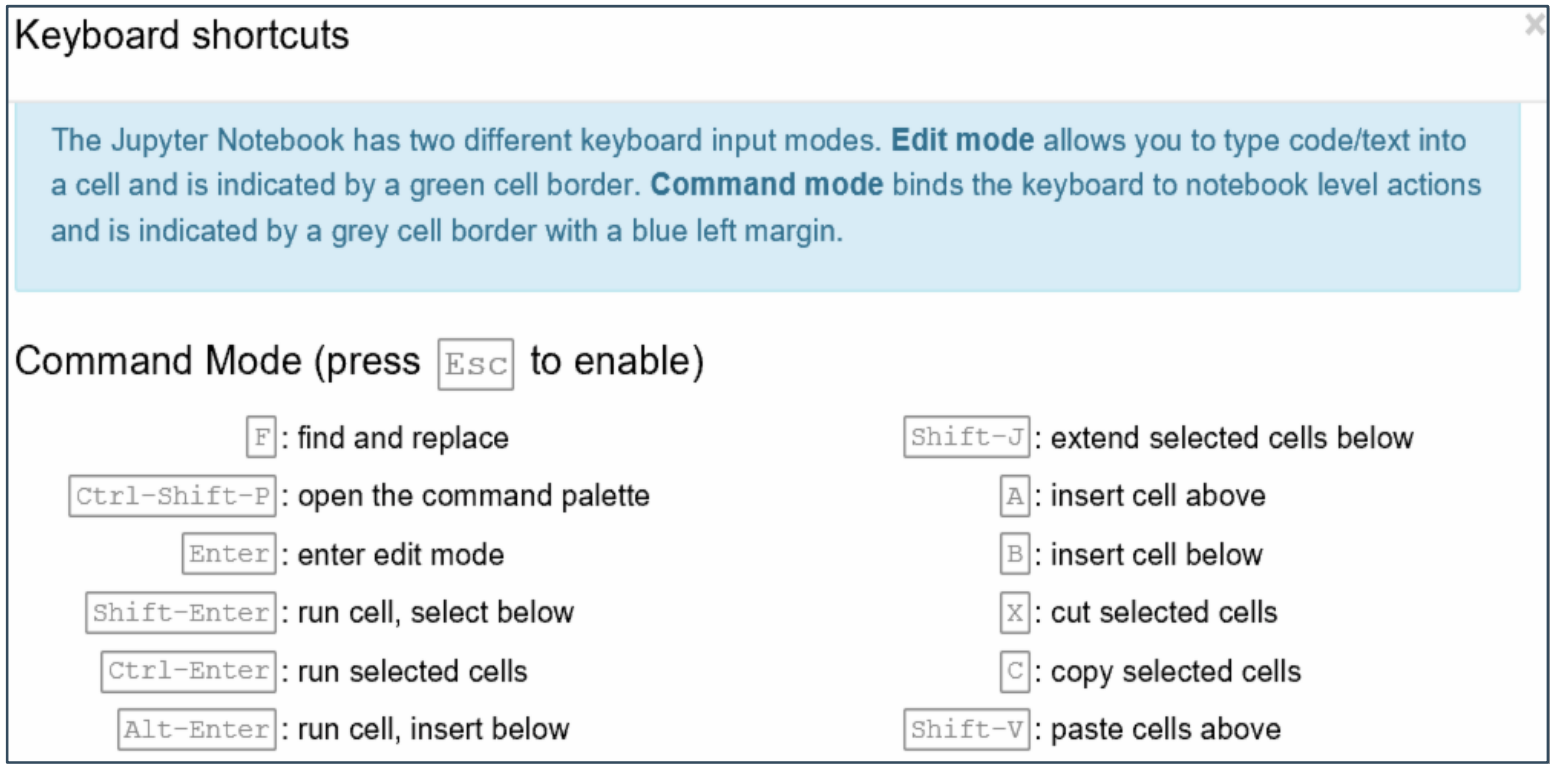
- %matplotlib inline
  - 将图表显示在Jupyter notebook中
- %%%timeit
  - 记录一个单元的执行时间
- %run filename.ipynb
  - 运行另一个notebook或python文件的代码
- %load filename.py
  - 将外部文件的内容拷贝粘贴到单元中

test.py

```
import caffe
SolverName = "/root/workspace“
sovler = caffe.AdamSolver(SolverName)
```



Jupyter快捷键



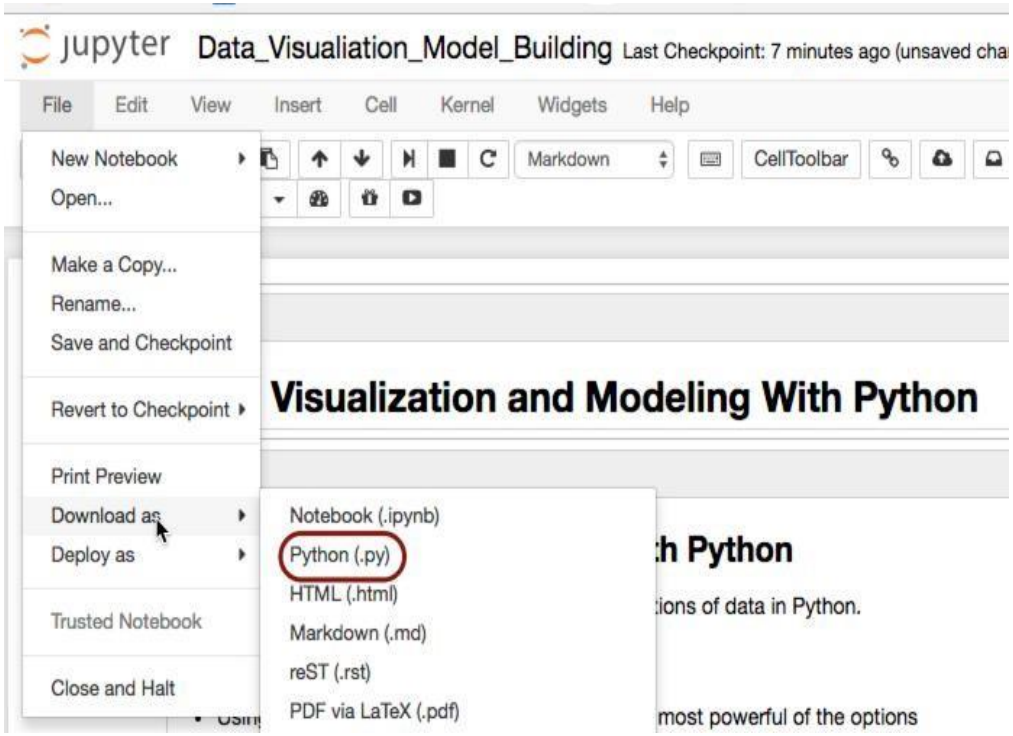
可以通过Help → Keyboard Shortcuts查看所有的快捷键

抽取Python代码

从命令行转换：

```
>>> jupyter nbconvert --to python notebook.ipynb
```

从Notebook导出代码：



Jupyter Notebook参考资料

- Jupyter官网： https://jupyter.org/
- Jupyter Notebook快速入门：
  - http://codingpy.com/article/getting-started-with-jupyter-notebook-part-1/
  - http://codingpy.com/article/getting-started-with-jupyter-notebook-part-2/
- Markdown语法说明：
  - https://www.appinn.com/markdown/

Python基本语法

- 一个语句占一行
  - 单个语句占多行，用反斜杠\
  - 多个语句在一行，使用分号;分隔
- Python使用缩进表示代码块，而不是一对花括号{}
- 注释：
  - 单行注释用#
  - 多行注释用三个单引号 ''' 或者三个双引号 """ 将注释括起来

Hello World!

- print函数：

```
print(“Hello World!”)
```

数据类型（1）

- 基本数据类型：
  - 数字（number）
    - 整数（int）：*2，59，100，-3*
    - 小数（float）：*0.4，5.0，-0.78*
  - 布尔（bool）：*True，False*
  - 字符串（str）：*”How are you?”，’this is a string.’*

数据类型（2）

- 容器类型:
  - 列表（list）：*[1, 2, 5, 10]*
  - 元组（tuple）：*(1, ‘two’)*
  - 集合（set）：*{‘Mike’, ‘John’, ‘Marry’}*
  - 字典（dictionary）：*{‘20120010’: 98, ‘20120011’: 89, ‘20120023’:100}*

变量

- 变量在被赋值的时候创建，无需声明
- 变量可以在任何时候被重新赋值为任何其他类型的值

```
message = “Hello World!”
print(message)

message = “Hello Python Course!”
print(message)
type(message)

message = 3.6
print(“Python “, message)
type(message)

message = 100
print(“There are “, message, “cars”)
type(message)
```

```
a, b, c, d = 20, 5.5, True, 4+3j
print(type(a), type(b), type(c), type(d))
```

变量命名规则

- 变量名只能包含字母、数字和下划线。
- 变量名只能以字母或下划线开头。
- 变量名是大小写敏感的。
- 变量名不要使用Python的关键字或函数名

下面哪些变量名是不合法的：
current balance, current_balance, currentBalance, current-balance, 4account, account4, _spam, _Spam, print, ‘hello’

算术表达式

- 基本的算术运算符：+、-、*、/、//、%、**

```
5 + 4     # 加法
4.3 - 2   # 减法
3 * 7     # 乘法
2 / 4     # 除法，得到浮点数
2 // 4    # 除法，得到整数
17 % 3    # 取余
2 ** 5    # 乘方
```

```
x = 3
print(type(x)) # Prints "<class 'int'>"
print(x)
print(x + 1)
print(x - 1)
print(x * 2)
print(x ** 2)

x += 1
print(x)
x *= 2
print(x)

y = 2.5
print(type(y)) # Prints "<class 'float'>"
print(y, y + 1, y * 2, y ** 2)
```

布尔表达式

- ‘True’ 和 ‘ False’为预定义值；实际上是整数1 和 0
- 比较运算符：<、<=、>、>=、==、!=，结果是布尔值
- 布尔运算符：not、and、or

```
12 < 13

12 > 13

12 <= 12

12 != 13

True * 12

0 and 1

(3+2) < (5-7)

(7==6) and (12!=21)
```

```
t = True
f = False

print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND;
print(t or f)  # Logical OR;
print(not t)   # Logical NOT;
print(t != f)  # Logical XOR;
```

字符串

- 字符串用单引号'或双引号"括起来
- 使用反斜杠\转义特殊字符：\n 换行符，\t 制表符，……
- 三引号用于大块的文本内容

```
a = 'Hello world!'
b = "Hello world!"
a == b

a = "Per's lecture"
print(a)
```

```
a = "One line.\nAnother line."
print(a)

b = """One line,
another line."""
print(b)
```

字符串运算

```
# 字符串拼接
a = "Part 1"
b = "and part 2"
a + ' ' + b

# 字符串重复并拼接
s = a * 2
print(s)

# 提取子串
s[0]
print(s[0:4])
print(s[5:])
print(s[6:-1])

# 字符串长度
len(s)

# 子串检测
'p' in s
'P' in s
'Part' in s
```

字符串不能改变

- Python中的字符串不可修改！

```
# 直接修改字符串会报错
s[0] = 'B'
```

- 如果要改变一个字符串：用旧的字符串片段生成一个新的

```
# 构建新字符串
s = 'B' + s[1:]
```

- 如果要生成许多新串，尝试字符串格式化

```
hello = 'hello'
world = "world"
# 字符串格式化
hw12 = '%s %s %d' % (hello, world, 12)
print(hw12)
```

- 列表（List）处理能让字符串处理更为有效

字符串的方法

- 字符串有一组内建（built-in）方法
- 没有方法可以改变原串，有几个方法可以生成新串

```
s = 'a string, with stuff'
s.count('st')                          # 有多少子串？
s.find('stu')                          # 寻找子串，如果有，给出子串的位置
s.replace('stuff', 'characters')       # 替换子串 (全部出现过的子串)
s.replace('s', 'X', 1)                 # 只替换一次

s = '3'
s.isdigit()                            # 是纯数字串吗？
```

```
s = "hello"
print(s.capitalize())          # 首字母大写；输出"Hello"
print(s.upper())               # 所有字符转换成大写字符；输出"HELLO"
print(s.rjust(7))              # 右对齐，左端补空格；输出"  hello"
print(s.center(7))             # 居中对齐，左右两端补空格；输出" hello "

print('  world '.strip())      # 去除前后的所有空白符；输出"world"
```

列表

- 有序的对象序列
- 异质的；可以包含任意类型的对象的混合

```
r = [1, 2.0, 3, 5]          # 列表实例，不同的值
type(r)                     # 输出<class 'list'>
```

```
r[1]                        # 通过下标来访问；偏移量为 0
r[-1]                       # 负的下标代表从尾部开始计数
r[1:3]                      # 列表的片段；给出新的列表
```

```
w = r + [10, 19]            # 合并列表；给出另外的一个列表
w
r                           # 原列表不变；w 和 r 不同
```

```
t = [0.0] * 10              # 用重复生成一个初始向量
t
```

列表操作

- 列表是可变的，可以改变局部

```
r = [1, 2.0, 3, 5]
r[3] = 'word'                    # 通过下标改变一个元素（项）
r                                # 显示[1, 2.0, 3, 'word']
```

```
r[0] = [9, 8]                    # 列表可以嵌套
r                                # 显示[[9, 8], 2.0, 3, 'word']
```

```
r[0:3] = [1, 2, 5, 6]            # 改变列表的一个片段，可以改变列表的长度
r                                # 显示[1, 2, 5, 6, 'word']

r[1:3] = []                      # 通过设置列表的片段为空集来移除元素
r                                # 显示[1, 6, 'word']
```

```
len(r)                           # 列表的长度，即项的个数，显示3
```

```
6 in r                           # 成员测试，显示True
r.index(6)                       # 搜索并给出位置，如果没有的话，报错，这里显示1
```

列表的方法（1）

```
r = [1, 2.0, 3, 5]
r.append('thing')                    # 在列表尾增加一个项
r                    # 显示[1, 2.0, 3, 5, 'thing']

r.append(['another', 'list'])        # 增加的列表被看作一个单一项
r                    # 显示[1, 2.0, 3, 5, 'thing', ['another', 'list']]
```

```
r = [1, 2.0, 3, 5]
r.extend(['item', 'another'])  # 列表的项逐次添加
r                    # 显示[1, 2.0, 3, 5, 'item', 'another']
```

```
k = r.pop()                          # 移除最后一项
k                                    # 显示'another'
r                                    # 显示[1, 2.0, 3, 5, 'item']

r.insert(3, 4.0)                     # 在指定位置插入一项
r                                    # 显示[1, 2.0, 3, 4.0, 5, 'item']

r.remove('item')                     # 删除一项
r                                    # 显示[1, 2.0, 3, 4.0, 5]
```

列表的方法（2）

- 使用内建的sort方法：排序是内部进行的，不产生新列表!
- 外部函数sorted，不改变原列表的顺序
  - https://blog.csdn.net/java276582434/article/details/90812971

```
r = [2, 5, -1, 0, 20]
r.sort()
r                                    # 显示[-1, 0, 2, 5, 20]

w = ['apa', '1', '2', '1234']
w.sort()                             # 字符串：使用ASCII顺序
w                                    # 显示['1', '1234', '2', 'apa']
```

```
w.reverse()                          # 反转列表!
w                                    # 显示['apa', '2', '1234', '1']

v = w[:]                             # 首先生成新表
v.reverse()                          # 反转这份拷贝
v                                    # 显示['1', '1234', '2', 'apa']
w                                    # 显示['apa', '2', '1234', '1']
```

转换字符串为列表

```
s = 'biovitrum'              # 生成字符串
w = list(s)                  # 转为字符的列表
w                      # 显示['b', 'i', 'o', 'v', 'i', 't', 'r', 'u', 'm']
w.reverse()
w                      # 显示['m', 'u', 'r', 't', 'i', 'v', 'o', 'i', 'b']

r = ''.join(w)               # 使用空串的join方法
r                            # 显示'murtivoib'

d = '-'.join(w)              # 使用字符-的join方法
d                            # 显示'm-u-r-t-i-v-o-i-b'

s = 'a few words'
w = s.split()                # 基于空白符(空格, 新行)切分
w                            # 显示['a', 'few', 'words']

' | '.join(w)               # 对其他串用方法'join', 显示'a | few | words'
```

元组

- 和列表一样，除了不可变，即一旦生成，就不可改变
- 某些函数会返回元组

```
t = (1, 3, 2)
t[1]                                    # 由下标访问，偏移量从0开始，显示3

(a, b, c) = t                           # 元组赋值
a                                       # 显示1
b                                       # 显示3

a, b, c                                 # 一个实际上的元组表达式!显示(1, 3, 2)
```

```
a, b = b, a                             # 交换值的技巧
a, b                                    # 显示(3, 1)
```

```
r = list(t)                             # 转换元组为列表
r                                       # 显示[1, 3, 2]

tuple(r)                                # 转换列表为元组，显示(1, 3, 2)
```

集合

- 一个**无序**的**没有重复元素**的序列
- 基本功能是进行成员关系测试和删除重复元素，可以进行集合运算

```
student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}
print(student)                          # 输出集合，重复元素被自动去掉
{'Mary', 'Jim', 'Rose', 'Jack', 'Tom'}
print('Rose' in student)                # 成员测试
True

a = set('abracadabra')                  # 字符串转换成集合
b = set('alacazam')
print(a)
{'b', 'a', 'c', 'r', 'd'}
print(a - b)                            # a和b的差集
{'b', 'd', 'r'}
print(a | b)                            # a和b的并集
{'l', 'r', 'a', 'c', 'z', 'm', 'b', 'd'}
print(a & b)                            # a和b的交集
{'a', 'c'}
print(a ^ b)                            # a和b中不同时存在的元素
{'l', 'r', 'z', 'm', 'b', 'd'}
```

字典

- **键（key）值（value）**对的无序集合
- 键必须使用不可变类型。可以用数字、字符串或元组，不能用列表
- 在同一个字典中，键必须是唯一的，值不必

```
h = {'key': 12, 'nyckel': 'word'}
h['key']                    # 由键访问，显示12

h.has_key('nyckel')         # 显示True
```

- 增加一个键/值

```
h['Per'] = 'Kraulis'
h
{'nyckel': 'word', 'Per': 'Kraulis', 'key': 12} # 输出顺序是随机的
```

- 替换一个键对应的值

```
h['Per'] = 'Johansson'
h
{'nyckel': 'word', 'Per': 'Johansson', 'key': 12}
```

字典的方法（1）

```
h = {'key': 12, 'nyckel': 'word'}
'Per' in h                          # 测试一个键是否在字典中，显示False

h['Per']                            # 报错
```

```
h.get('Per', 'unknown')             # 返回值，或者返回缺省值，显示'unknown'

h.get('key', 'unknown')             # 显示12
```

```
h.keys()                            # 字典中所有的键
['nyckel', 'key']
h.values()                          # 字典中所有的值
['word', 12]
h.items()                           # 字典中所有的键值对
dict_items([('key', 12), ('nyckel', 'word')])
```

```
len(h)                              # 字典中键值对的个数，显示2
```

删除数据的命令：del

- 命令！不是函数！
- 实际上移除变量（名字），不是对象

```
a = 'thing'                    # 定义一个变量
a                              # 显示'thing'

del a                          # 把这个变量忘掉
a                              # 报错
```

```
h = {'key': 12, 'nyckel': 'word'}
del h['key']                   # 移除键和它的值
h                              # 显示{'nyckel': 'word'}
```

```
r = [1, 3, 2]
del r[1]                       # 另一个删除列表项的方式
r                              # [1, 2]
```

字典方法（2）

```
g = h.copy()                    # 拷贝字典

del h['key']
h                               # 显示{'nyckel': 'word'}
g                               # 显示{'nyckel': 'word', 'key': 12}

h['Per'] = 'Johansson'
h                               # 显示{'nyckel': 'word', 'Per': 'Johansson'}

h.update(g)     # 根据g添加或者更新所有的键值
h               # 显示{'nyckel': 'word', 'key': 12, 'Per': 'Johansson'}

g.clear()       # 清除字典中的所有项
print(len(g))   # 显示0
```

条件语句：if

if condition_1:
  statement_block_1
[**elif** condition_2:
  statement_block_2]
[**else:**
  statement_block_3]

```
age = 17
if age >= 18:
    print("You are old enough to vote!")
    print("Have you registered to vote yet?")
else:
    print("Sorry, you are too young to vote.")
    print("Please register to vote as soon as you turn 18!")
```

```
age = 12
if age < 4:
    print("Your admission cost is $0.")
elif age < 18:
    print("Your admission cost is $5.")
else:
    print("Your admission cost is $10.")
```

循环语句：while

while condition:
    statement_block

```
current_number = 1
while current_number <= 5:
    print(current_number)
    current_number += 1
```

```
pets = ['dog', 'cat', 'dog', 'goldfish', 'cat', 'rabbit', 'cat']
print(pets)

while 'cat' in pets:
    pets.remove('cat')
print(pets)
```

循环语句：for

for variable **in** sequence:
    statement_block
else:
    statement_block

```
magicians = ['alice', 'david', 'carolina']
for magician in magicians:
    print(magician.title() + ", that was a great trick!")
```

```
squares = []
for value in range(1,11):
    squares.append(value**2)
print(squares)
```

```
r = []
for c in 'this is a string with blanks': # 一个字符一个字符地遍历字符串
    if c == ' ': continue                # 跳过后面的代码块,继续循环
    r.append(c)
print (''.join(r))
```

循环中的break, continue和else

- break语句跳出循环
- continue语句结束本轮循环，开始下一轮循环
- else在循环条件不满足时被执行，被break的循环不执行else
- pass语句是空语句，什么都不做，占位语句

```
r = [1, 3, 10, 98, -2, 48]
for i in r:
    if i < 0:
        print ('input contains negative value!')
        break                                   # 跳出整个循环，包括'else'
    else:
        pass                                    # 什么都不做
else:                                           # 如果循环是正常结束的，则执行
    print ('input is OK')
```

列表循环与List Comprehension

```
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)                          # Prints [0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]
squares = [x ** 2 for x in nums]
print(squares)                          # Prints [0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)                     # Prints [0, 4, 16]
```

```
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
```

字典循环与Dictionary Comprehension

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))
```

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
```

```
nums = [0, 1, 2, 3, 4]
even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
print(even_num_to_square)                    # Prints {0: 0, 2: 4, 4: 16}
```

集合循环与Set Comprehension

```
animals = {'cat', 'dog', 'fish'}
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
```

```
from math import sqrt

nums = {int(sqrt(x)) for x in range(30)}
print(nums)                              # Prints {0, 1, 2, 3, 4, 5}
```

函数

def 函数名（参数列表）:
  函数体

```
def greet_user():
    print("Hello!")

greet_user()
```

```
def greet_user(username):
    print("Hello, " + username.title() + "!")

greet_user('jesse')
```

```
def describe_pet(animal_type, pet_name):
    print("\nI have a " + animal_type + ".")
    print("My " +animal_type+ "'s name is " + pet_name.title() + ".")

describe_pet('hamster', 'harry')
describe_pet('dog', 'willie')
```

函数参数的默认值

- 参数可以有默认值
- 当调用时没有给定参数，会采用默认值
- 有默认值的参数必须放在参数列表的最后
- 显式调用参数，可以改变参数顺序

```
def describe_pet(pet_name, animal_type='dog'):
    print("\nI have a " + animal_type + ".")
    print("My ‟ +animal_type+ "'s name is " +pet_name.title()+ ".")

describe_pet('willie’)
describe_pet(pet_name='harry', animal_type='hamster’)
describe_pet(animal_type='hamster', pet_name='harry')
```

函数返回值

- 一个函数不一定要有return语句
- 实际上，函数默认总会返回一个值：'None'
- 'None' 是一个特殊的值，意味着 '什么都没有'

```
def get_formatted_name(first_name, last_name):
    full_name = first_name + ' ' + last_name
    return full_name.title()

musician = get_formatted_name('jimi', 'hendrix')
print(musician)
```

模块

- 将函数存储在被称为模块的独立文件中，再将模块导入到主程序中
- 数学函数在一个单独的模块中

```
import math                              # 导入整个'math'模块

print (math.e, math.pi)
print (math.cos(math.radians(180.0)))
print (math.log(10.0))
print (math.exp(-1.0))
```

```
from math import *                       # 导入模块'math'中的所有函数

print (e, pi)
print (cos(radians(180.0)))
print (log(10.0))
print (exp(-1.0))
```

```
from math import log, cos                # 导入'math'模块中的log和cos函数
```

```
from math import log as lg               # 导入'math'模块中的log函数，起别名lg
```

文件操作：读

- 一个文件操作对象由内建函数 'open' 创建
- 文件对象有一系列函数
- ‘read’：读取整个文件 （或者说N 字节），返回一个单独的字符串
- ‘readline’：读取一行（然后跳到新的一行）
- ‘readlines’：读取所有的行，返回一个字符串的列表

```
f = open('test.txt')    # 默认: 只读模式

line = f.readline()     # 读一行
line                    # 显示'This is the first line.\n’

lines = f.readlines()   # 读所有剩余行
lines                   # 显示['This is the second.\n', 'And third.\n']
```

文件操作：写

- ‘write’ 函数只是简单地输出给定字符串
- 字符串不一定是ASCII码，二进制串也可以

```
w = open('output.txt', 'w')          # 写模式 （默认写的是文本）

w.write('stuff')                     # 并不自动添加新行
w.write('\n')
w.write('more\n and even more\n’)

w.close()
```

```
stuff
more
 and even more
```

用for循环读取文件

```
infile = open('test.txt')                    # 只读模式
outfile = open('test_upper.txt', 'w')        # 写模式; 创建文件

for line in infile:                          # 遍历文件中的每一行
    outfile.write(line.upper())

infile.close()                               # 并不严格要求; 系统会自动执行
outfile.close()
```

- 注意: 每行结尾会尾随一个换行符 ‘\n’
- 可以使用字符串方法’strip’或者’rstrip’去除它

参考资料和教程

- Python文档：https://docs.python.org/3/
- “用Python玩转数据”：
  https://www.coursera.org/learn/hipython/
- **《Python编程：从入门到实践》**
  - 第2、3、4、5、6、7、8章