Python基础

孙鹰

前言：Python哲学

- Python/Lib/idlelib/idle.bat，输入 import this
- Beautiful is better than ugly. 优美胜于丑陋
- Explicit is better than implicit. 明了胜于晦涩
- Simple is better than complex. 简单胜过复杂
- Complex is better than complicated. 复杂胜过凌乱
- Flat is better than nested. 扁平胜于嵌套
- Sparse is better than dense. 间隔胜于紧凑
- Readability counts. 可读性很重要
- In the face of ambiguity, refuse the temptation to guess. 当存在多种可能时，不要尝试去猜测

前言： Python哲学

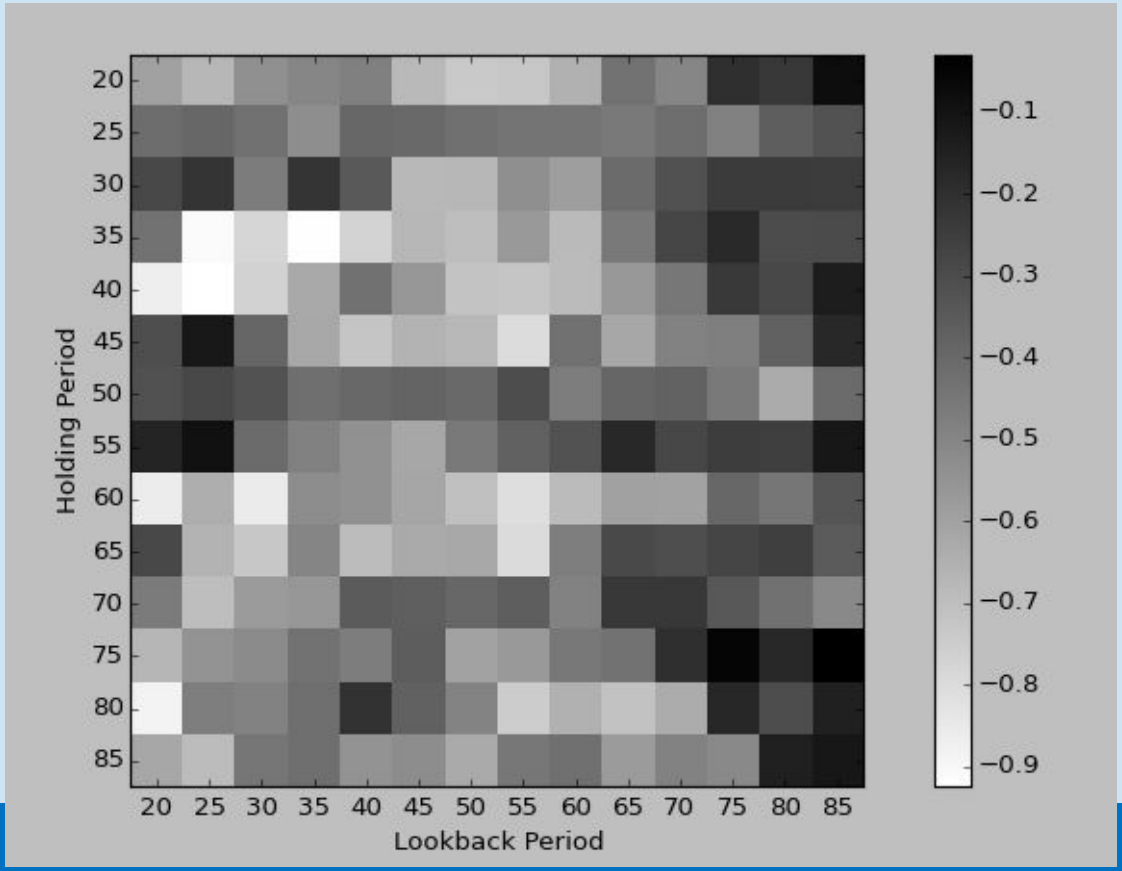
- 结论： C语言能够做到的，我们也能做到，而且能够做的更简单更优美。
- 抛弃旧的习惯，使用新的语法。
- Eg:
- 1. a<b<c
- 2.a+bj
- 3.[]
- 4.特别的分隔符

前言： Python作用

- 绘制你想象不到的复杂图表
- 股票收益累计率

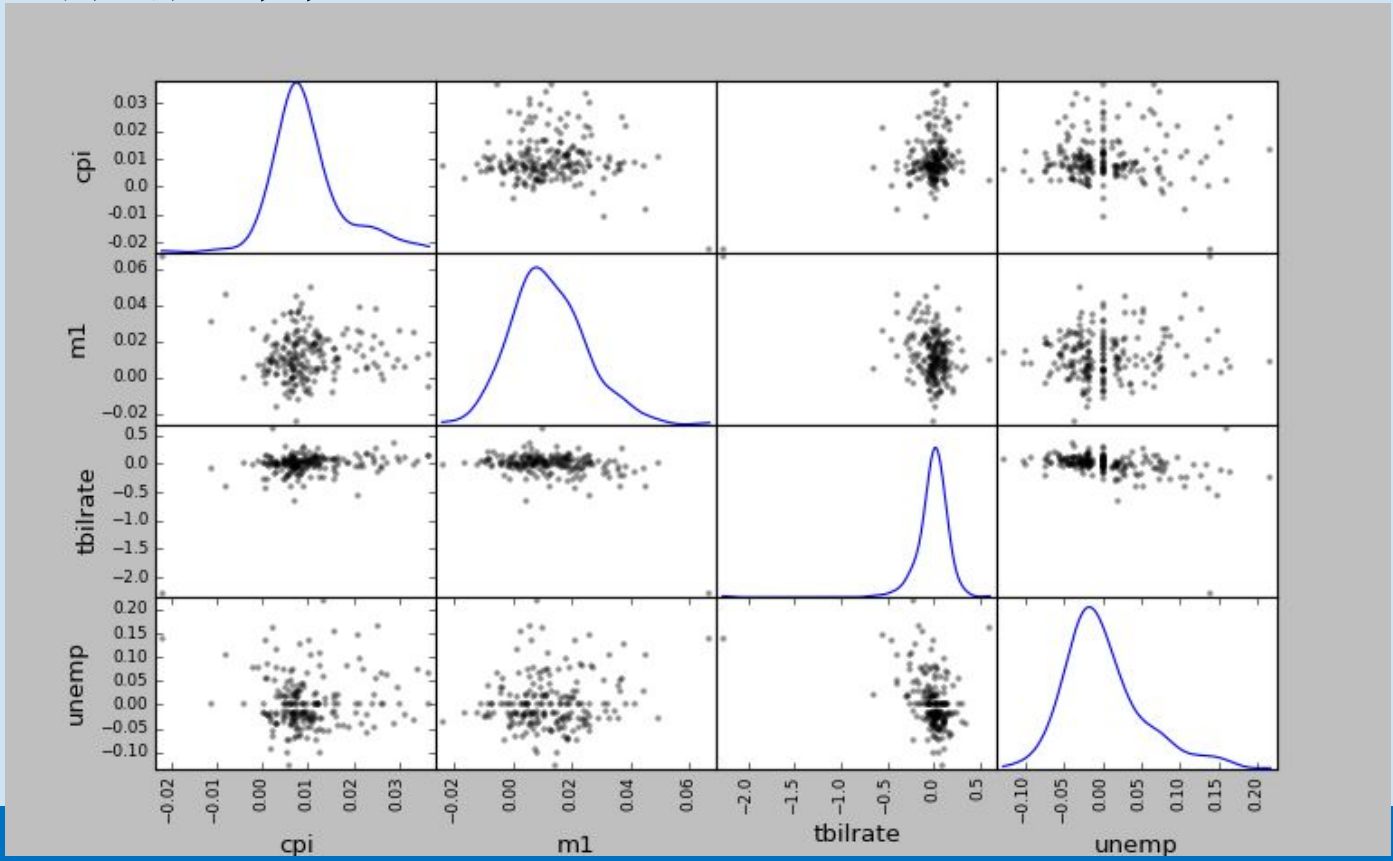
前言：Python作用

- 绘制你想象不到的复杂图表
- 动量策略回顾起与持有期比例热图



前言：Python作用

- 绘制你想象不到的复杂图表
- 散列图矩阵



前言： Python作用2

- 深度学习（Deep learning）

前言： Python作用2

- 深度学习（Deep learning）

前言： Python作用2

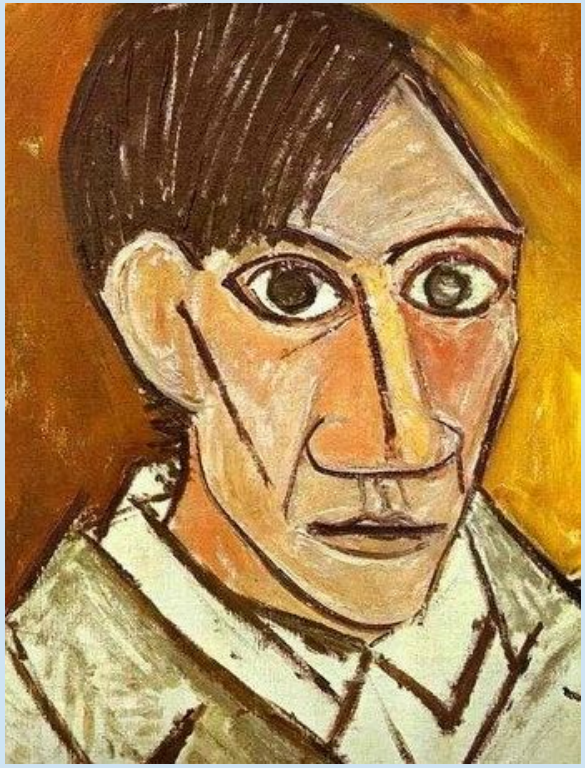
- 深度学习（Deep learning）

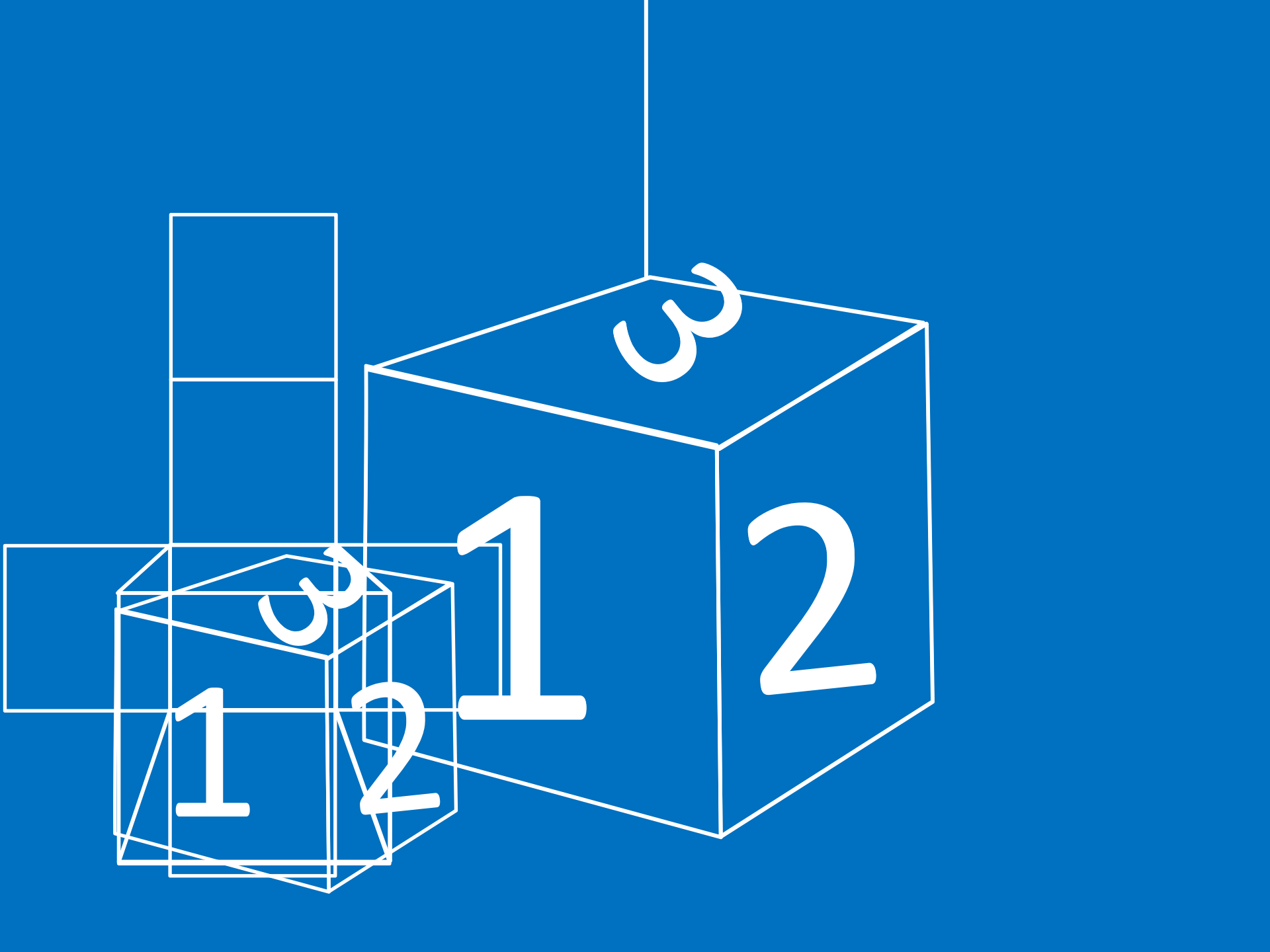
前言： Python作用2

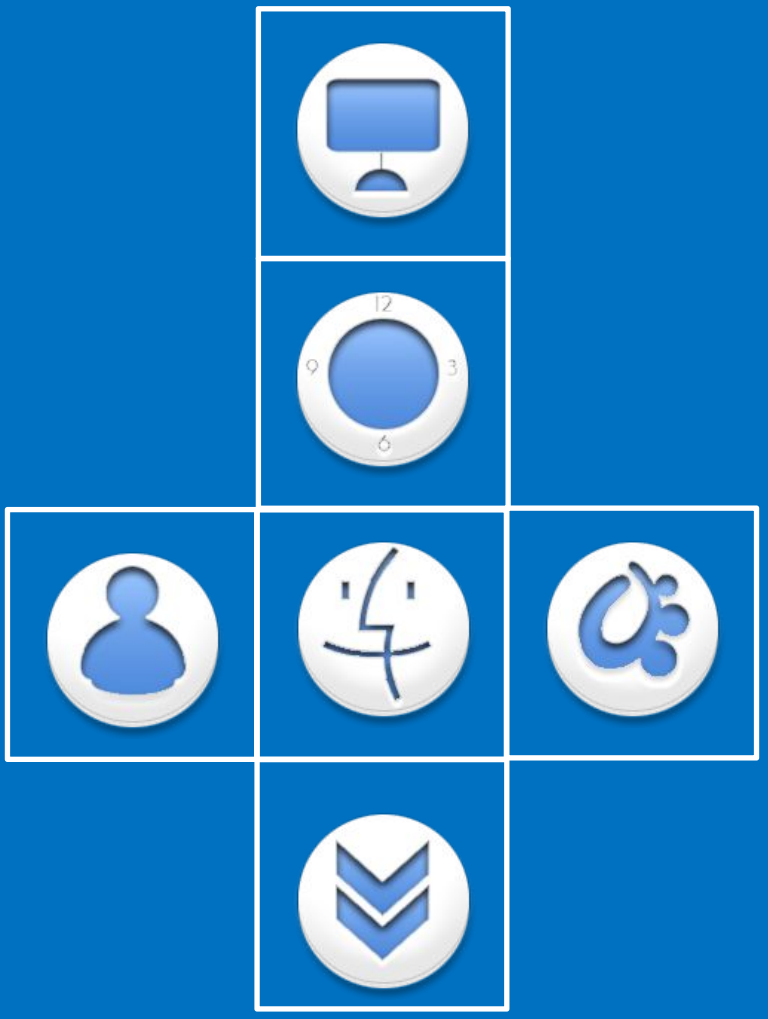
- 深度学习（Deep learning）

前言： Python作用2

- 可怕的开源力量，请猜哪个是我的同学
- https://github.com/jcjohnson/neural-style




3
1
2
3
1
2
1



- 输入输出
- 数据类型
- 控制结构
- 函数定义
- 异常处理
- 面向对象

输入输出

Index

- 输入
- 输出
- 格式化字符串
- 文件

输入

- 回忆：图灵机。
- 在程序运行过程中需要与用户进行交互以获取数据或提供结果
- 输入函数input([prompt])
- 并不推荐使用raw_input()，最好忘记这茬。

输入

- **In [1]:** name=input("Please enter your name: ")
- Please enter your name: SY
- NameError: name 'SY' is not defined

- **In [2]:** name=input("Please enter your name: ")
- Please enter your name: 'SY'

- **In [3]:** name
- **Out[3]**: 'SY‘
- **In [4]: type(name)**
- **Out[4]: str**

输入

- In [5]: age=input("Please enter your age: ")
- Please enter your age: 18

- In [6]: type(age)
- Out[6]: int

输出

- 输出函数print()
- 原型： print(*object, sep=‘ ’, end=‘\n’,
- file=sys.stdout)
- Python2.7 中：
- from __future__ import print_function

输出

- **In [12]: print("Hello", "World", sep="###",end='***')**
- **Hello###World***

- **In [13]: print("Hello", "World")**
- **Hello World**

格式化字符串

- **回忆：C语言中printf和scanf，在python中如何实现？**
- 格式化字符串
- 字符串模板 + % +元组/字典
- %称为格式操作符，和转换字符一起，指明了数据类型

格式化字符串

符号	类型
d/i	整型
u	无符号整型
f	浮点型，m.ddddd
e	浮点型，m.dddddde+/-XX
E	浮点型，m.ddddE+/-XX
g	指数比-4小或比5大时使用%e，否则使用%f
c	单字符
s	字符串或者能通过str()转为字符串的数据
%	输入一个%

修饰符类型	说明
num	是该值占据num个字符宽度
-num	num个字符宽度，左对齐
+num	num个字符宽度，右对齐
0num	num个字符宽度，前置“0”
.num	保留num位小数
(name)	从字典中取key为name的值放在该处

格式化字符串

- **In [16]: price=55;item='banana'**
- **In [17]: "The %s costs %d cents" % (item, price)**
- **Out[17]: 'The banana costs 55 cents‘**
- **In [19]: "The %(item)s costs %(cost)7.1f cents" % itemdict**
- **Out[19]: 'The banana costs 55.0 cents'**

文件

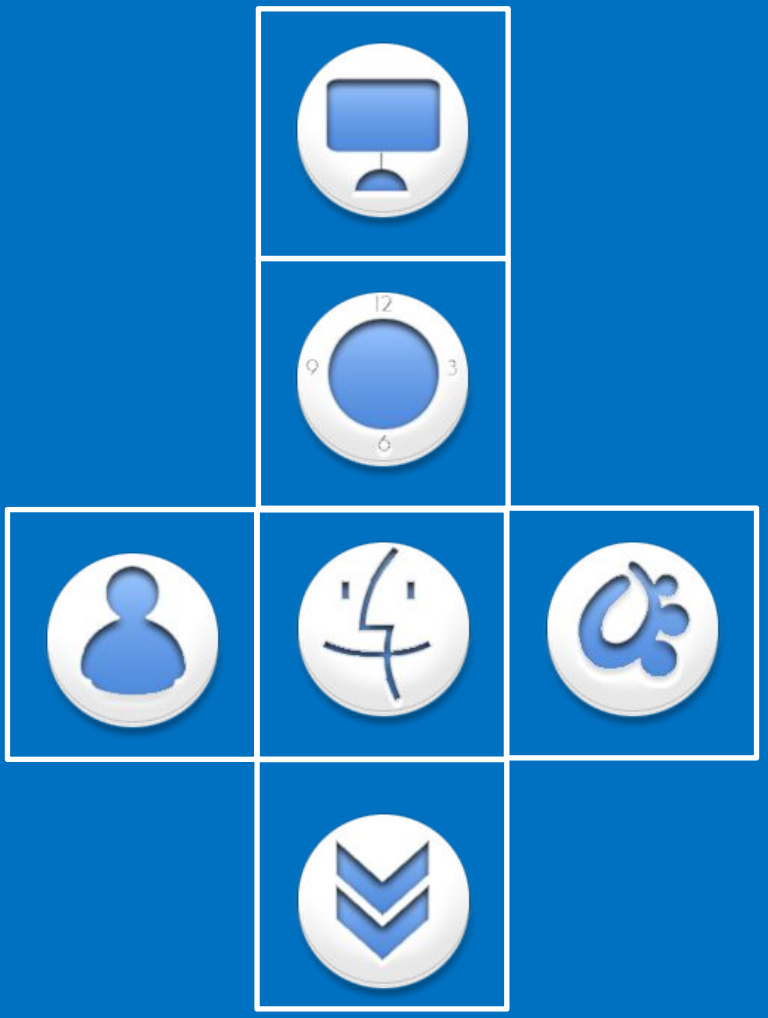
- open(name[,mode[,buffering]])
  name参数表示需要打开的文件名称，mode是打开模式，open()函数的第三个参数用来控制文件的缓冲。
- w 以写方式打开
- r 以读方式打开
  a 以追加模式打开 (从 EOF 开始, 必要时创建新文件)
  r+,w+,a+ 以读写模式打开
  rb 以二进制读模式打开
  wb 以二进制写模式打开 (参见 w)
  ab 以二进制追加模式打开 (参见 a)
  rb+,wb+,ab+ 以二进制读写模式打开

文件

- In [27]: f=open("D:\\test.txt", "w")
- In [28]: f.write("function write\n")
- >>>strList=["Hello\n", "World\n"]
- >>>f.writelines(strList)
- In [29]: f.close()

- In [30]: f=file("D:\\test.txt", "r")
- ...: f.readline()
- Out[30]: 'function write\n'

- In [31]: strList=f.readlines()
- In [32]: strList
- Out[31]: ['Hello\n', 'World\n']



- 输入输出
- 数据类型
- 控制结构
- 函数定义
- 异常处理
- 面向对象

基本类型

Index

- 赋值过程
- 强类型
- 基础类型
- 逻辑判断

赋值过程

- Python中的变量名以字母或下划线开始，大小写敏感，可以是任意长度
- 给变量取一个有意义的名字是编程的良好习惯，方便自己和他人阅读理解
- 当一个名字第一次被用于赋值符号（=）左边时，一个python变量就被创建
- 赋值过程实际上是让变量保持实际数据的一个引用，而不是持有数据本身，该思想类似于C语言中的指针概念

赋值过程

Figure 3: Variables Hold References to Data Objects

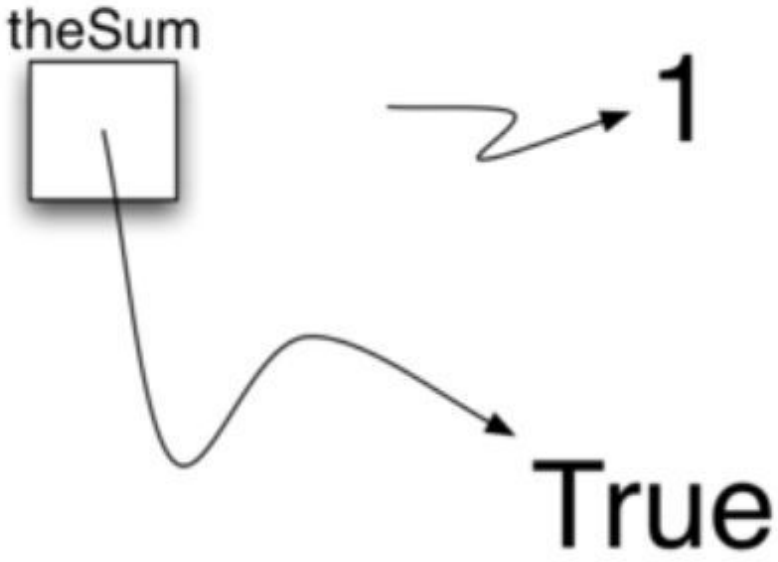


Figure 4: Assignment Changes the Reference

赋值过程

- In [59]: theSum=0
- ...: >>> theSum
- Out[59]: 0
- In [60]: id(theSum)
- Out[60]: 38759504L

- In [61]: theSum=theSum+1
- ...: >>> theSum
- Out[61]: 1
- In [62]: id(theSum)
- Out[62]: 38759480L

- In [63]: theSum=True
- ...: >>> theSum
- Out[63]: True
- In [64]: id(theSum)
- Out[64]: 505971488L

强类型

- 如果你认为python是一种非强类型的语言，那就错了。
- 思考：5+’5’=？
- VB中？JS中？
- In [66]: a=5
- ...: isinstance(a,float)
- Out[66]: False

- In [67]: a=5
- ...: isinstance(a,(float,int))
- Out[67]: True

基础类型

- 常见数据类型：
- Int
- Float
- 常见运算方式：
- +、-、*、/、%
- 地板除//，亦即floor()，相反的是ceil()
- 乘方**

逻辑判断

- 布尔型：
- True/False
- 逻辑操作符：
- and、 or、 not（特点：只能是英文单词）

逻辑判断

Operation	Meaning
<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal
is	object identity
is not	negated object identity

- 特别注意区分==和is

逻辑判断

- a=5;b=5

- In [51]: a is b
- Out[51]: True（注意！）
- In [52]: a == b
- Out[52]: True

- In [54]: c=5.0

- In [55]: a==c
- Out[55]: True
- In [56]: a is c
- Out[56]: False

特殊结构

Index

- 元组
- 列表
- 集合
- 字典
- 字符串
- 内置序列函数

元组

元组

- 回忆：C语言数组
- 元组是一维、定长、不可变的python对象序列。注意，它是一种序列。
- **In [72]: tup = 4, 5, 6**
- **...: tup**
- **Out[72]: (4, 5, 6)**

- **In [73]: nested_tup = (4, 5, 6), (7, 8)**
- **...: nested_tup**
- **Out[73]: ((4, 5, 6), (7, 8))**

元组

- 你能想得到的转换方式都是允许的
- **In [74]: tuple([4, 0, 2])**
- **Out[74]: (4, 0, 2)**

- **In [75]: tuple('string')**
- **Out[75]: ('s', 't', 'r', 'i', 'n', 'g')**
- **Python哲学：关键字大都是转换函数。**

元组

- tup = tuple(['foo', [1, 2], True])
- tup[2] = False

- tup[1].append(3)

猜猜可以不可以：

(4, None, 'foo') + (6, 0) + ('bar',)

('foo', 'bar') * 4

元组

- 拆包：梦想不再是梦想
- **In [76]: tup = 4, 5, (6, 7)**
- **...: a, b, (c, d) = tup**
- **...: d**

- **Out[76]: 7**

列表

列表

- 回忆：C语言链表
- 列表list是0个或多个python数据对象引用的有序容器，用“[]”或list函数构造

列表

- 列表是序列的，它支持许多操作。 这些操作对于其他任何python序列都适用
- []：索引
- +：连接
- *：重复
- in：某一项是否在序列中
- len()：获取序列的长度
- [:]：切片操作

列表

- In [77]: tup = ('foo', 'bar', 'baz')
- ...: b_list = list(tup)
- ...: b_list[1] = 33
- ...: b_list
- Out[77]: ['foo', 33 'baz']

列表

- In [81]: seq = [7, 2, 3, 7, 5, 6, 0, 1]
- ...: seq[1:5]
- Out[81]: [2, 3, 7, 5]

- In [82]: seq[:5]
- ...: seq[3:]
- Out[82]: [7, 5, 6, 0, 1]

- 一种巧妙的倒序方法：
- In [83]: seq[::-1] （如果是-2）
- Out[83]: [1, 0, 6, 5, 7, 3, 2, 7]

列表

- 列表本身实现了许多方法，常用于构建数据结构，方法的调用通过“.”实现
- aList.append(item)：在列表末尾添加一个新项
- aList.insert(i, item)：在列表的某个位置插入一个项
- aList.pop()：移除并返回列表的最后一项
- aList.pop(i)：移除并返回列表的第i项
- aList.sort()：对列表进行排序
- aList.reverse()：反转列表
- aList.index(item)：返回列表中第一个等于item项的索引
- aList.count(item)：返回列表中有多少项的值等于item
- aList.remove(item)：删除列表中第一个值等于item的项

列表

- In[77]:b_list=['foo', 'peekaboo', 'baz']

- In [78]: b_list.append('dwarf')
- ...: b_list
- Out[78]: ['foo', 'peekaboo', 'baz', 'dwarf']

- In [79]: b_list.insert(1, 'red')
- ...: b_list
- Out[79]: ['foo', 'red', 'peekaboo', 'baz', 'dwarf']

- In [80]: 'dwarf' in b_list
- Out[80]: True

集合

集合

- 回忆：高中集合
- 理发师悖论，罗素悖论
- 集合是不允许重复记录，并且用大括号括起来的无序集。

集合

- In[99]: set([2, 2, 2, 1, 3, 3])
- Out[99]: {1, 2, 3}

- In[101]: {1,2,3}=={3,2,2,1}
- Out[101]: True

集合

- 集合操作
- in：判断一个元素是否在集合内
- |：返回两个集合的并集
- &：返回两个集合的交集
- -：返回差集
- <=：判断第二个集合是否为第一个集合的超集

集合

- 集合运算
- union(otherset)：同“|”操作
- intersection(otherset)：同“&”操作
- difference(otherset)：同“-”操作
- insubset(otherset)：同“<=”操作
- add(item)：添加一个项
- remove(item)：移除一个项
- pop()：任意移除一个元素
- clear()：清空所有元素

集合

- **In [103]: a = {1, 2, 3, 4, 5};b = {3, 4, 5, 6, 7, 8}**

- **In [104]: a|b**
- **Out[104]: {1, 2, 3, 4, 5, 6, 7, 8}**

- **In [105]: a&b**
- **Out[105]: {3, 4, 5}**

- **In [106]: a-b**
- **Out[106]: {1, 2}**

- **In [107]: a^b**
- **Out[107]: {1, 2, 6, 7, 8}**

- **In [108]: a.union(b)**
- **Out[108]: {1, 2, 3, 4, 5, 6, 7, 8}**

集合

- 可变性
- **In [113]: a.add(4)**

- **In [114]: a**
- **Out[114]: {1, 2, 3, 4, 5}**

- **In [115]: a.remove(4)**

- **In [116]: a**
- **Out[116]: {1, 2, 3, 5}**

字典

字典

- 字典dict是key-value键值对的容器，又叫哈希映射（hash map）或相连数组（associative array）
- 字典用大括号括起来，每对元素为key:value
- 可以通过键来访问字典中的值，也可以添加一个新的键值对

字典

- In [119]: empty_dict = {}
- ...: d1 = {'a' : 'some value', 'b' : [1, 2, 3, 4]}
- ...: d1[7] = 'an integer'

- In [120]: d1
- Out[120]: {7: 'an integer', 'a': 'some value', 'b': [1, 2, 3, 4]}

- In [121]: 'an integer' in d1
- Out[121]: False
- In [122]: 'b' in d1
- Out[122]: True

字典

- **几乎包含所有list的方法，同时带有一些新增的方法：**
- keys()：返回一个list，包含所有键
- values()：返回一个list，包含所有值
- items()：返回一个list，包含所有键值对
- get(k)：返回键为k的值，如果不存在，返回None
- get(k, alt)：返回键为k的值，如果不存在，返回alt

字符串

字符串

- 字符串string是0个或者多个字符的序列容器
- 使用引号包围，可以是单引号，也可以是双引号，它是不可变的，多行需要三重引号。
- **In [132]: c = """**
- **...: This is a longer string that**
- **...: spans multiple lines**
- **...: """**

- **In [133]: c**
- **Out[133]: '\nThis is a longer string that\nspans multiple lines\n'**

字符串

- 字符串也有属于自己的方法
- aStr.center(w)：返回一个字符串，w长度，原字符串居中
- aStr.count(item)：返回原字符串中出现item的次数
- aStr.ljust(w)：返回一个字符串，w长度，原字符串居左
- aStr.rjust(w)：返回一个字符串，w长度，原字符串居右
- aStr.lower()：返回一个字符串，全部小写
- aStr.upper()：返回一个字符串，全部大写
- aStr.find(item)：查询item，返回第一个匹配的索引位置
- aStr.split(schar)：以schar为分隔符，将原字符串分割，返回一个列表
- 思考：aStr.replace(old, new[, count])

字符串

- In [134]: a = 'this is a string'
- ...: b = a.replace('string', 'longer string')
- ...: b
- Out[134]: 'this is a longer string‘

- In [139]: a.split('i')
- Out[139]: ['th', 's ', 's a str', 'ng']

- In [140]: a.split('is')
- Out[140]: ['th', ' ', ' a string']

- 思考：a[3]=‘s’

总结

总结

	元组	列表	集合	字典
是否可变				
序列散列				
使用符号				

内置序列函数

内置序列函数

- Enumerate用于跟踪当前序列的索引，产生的结果可以转化为字典。
- **In [174]: some_list = ['foo', 'bar', 'baz']**
- **...: mapping = dict(enumerate(some_list))**
- **In [175]: mapping**
- **Out[175]: {0: 'foo', 1: 'bar', 2: 'baz'}**

内置序列函数

- Sorted可以将任何序列返回为一个有序列表
- In[177]: sorted([7, 1, 2, 6, 0, 3, 2])
- Out[178]: [0, 1, 2, 2, 3, 6, 7]

- 下面这是干什么：
- **In [177]: sorted(set('this is just some string'))**
- **Out[177]: [' ', 'e', 'g', 'h', 'i', 'j', 'm', 'n', 'o', 'r', 's', 't', 'u']**

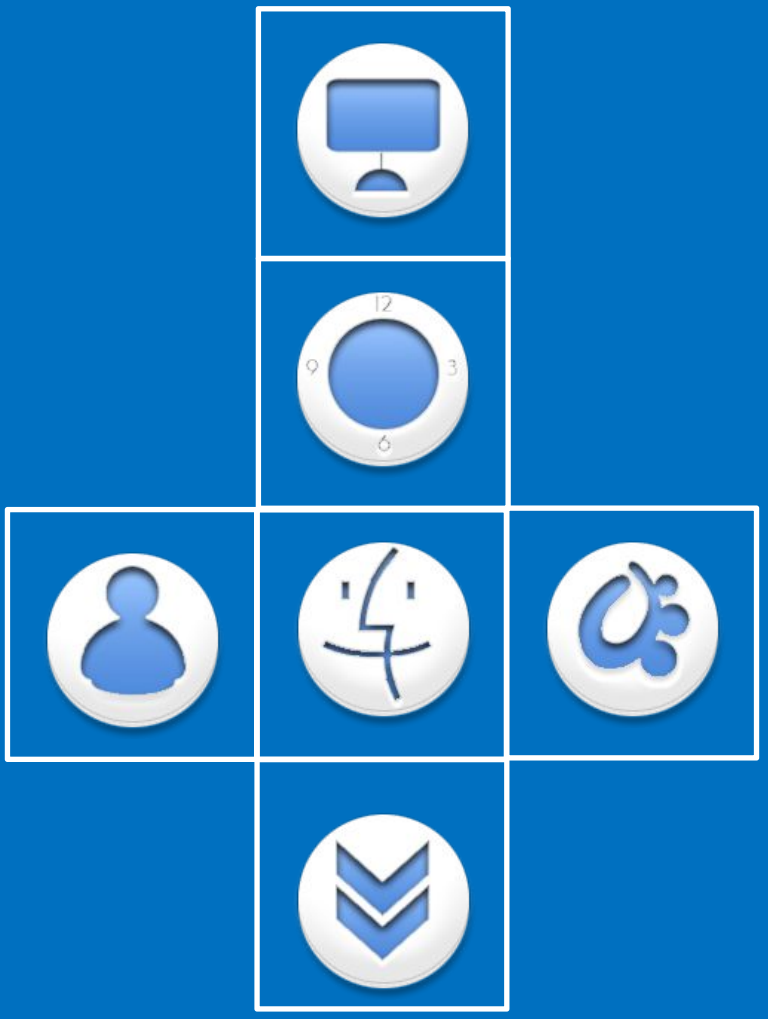
内置序列函数

- Zip用于将多个序列进行配对，生成元组的列表。也可以进行拆分基友配对。
- **In [179]: seq1 = ['foo', 'bar', 'baz']**
- **...: seq2 = ['one', 'two', 'three']**
- **...: zip(seq1, seq2)**
- **...:**
- **Out[179]: [('foo', 'one'), ('bar', 'two'), ('baz', 'three')]**

- **In [181]: pitchers = [('Nolan', 'Ryan'), ('Roger', 'Clemens'),**
- **...: ('Schilling', 'Curt')]**
- **...: first_names, last_names = zip(*pitchers)**
- **...: first_names**
- **Out[181]: ('Nolan', 'Roger', 'Schilling')**

内置序列函数

- Reversed用于逆序序列中的元素：
- **In [182]: list(reversed(range(10)))**
- **Out[182]: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]**



- 输入输出
- 数据类型
- 控制结构
- 函数定义
- 异常处理
- 面向对象

控制结构

Index

- 条件结构
- 循环结构
- 推导式

条件结构

- 回忆：C语言的if和else if
- In [143]: score = 33

- In [145]: if score >= 90:
- ...: print('A')
- ...: elif score >=80:
- ...: print('B')
- ...: elif score >= 70:
- ...: print('C')
- ...: elif score >= 60:
- ...: print('D')
- ...: else:
- ...: print('F')
-
- F
- 注意缩进即可

条件结构

- Python中没有switch-case结构
- Python中的三目操作符
- X if C else Y等价于C语言中的 C ? X : Y，更符合外国人的语言习惯。
- x = 3 if 4>=3 else 4
- x
- Out[148]: 3

循环结构

- 回忆：C语言for(int i = 0; i < n; i++)
- for语句用于遍历一个可迭代对象的成员，如列表、字符串等
- for语句结构如下，要强制自己使用！
- for item in collection:
- 具体到C语言的形式为：for i in range(n)
- While，continue，break用法和C语言一样

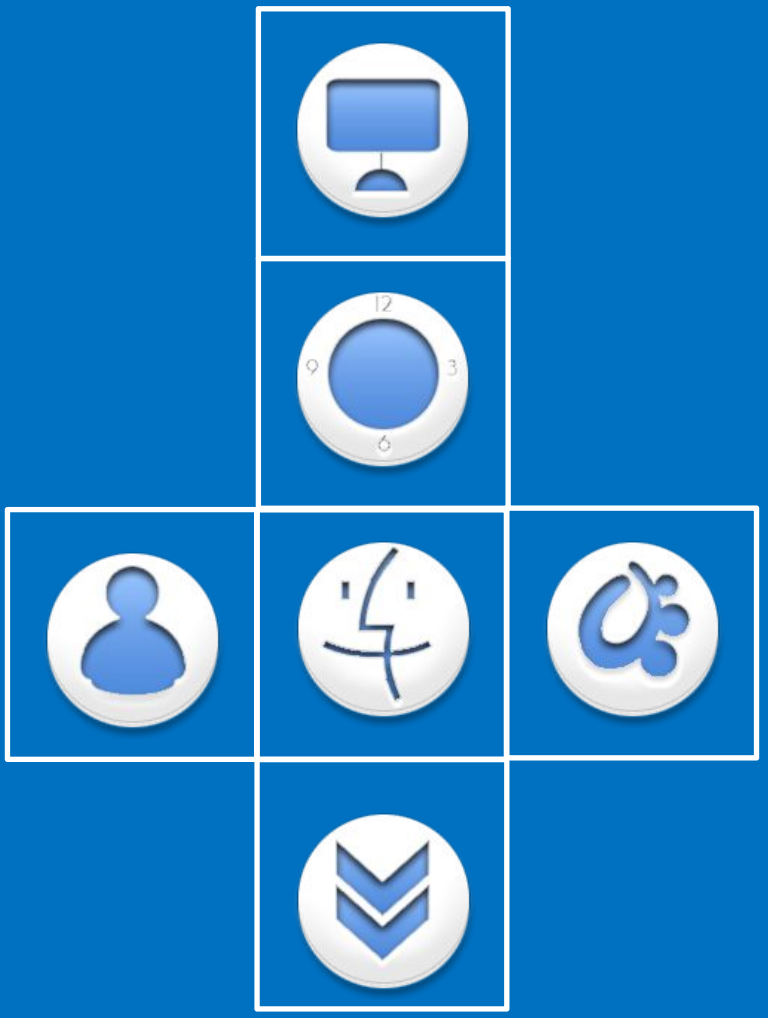
循环结构

- range()用于产生一组间隔平均的整数
- xrange()有同样的功能，只是不预先产生所有的值，用于数值巨大的时候。
- For远远不止于C中的for
- **In [146]: wordlist = ['cat','dog','rabbit']**
- **...: letterlist = []**
- **...: for aword in wordlist:**
- **...: for aletter in aword:**
- **...: letterlist.append(aletter)**
- **...: letterlist**
- **Out[146]: ['c', 'a', 't', 'd', 'o', 'g', 'r', 'a', 'b', 'b', 'i', 't']**

推导式

- 也叫解析式，为了方便地构建列表、字典等，将条件与表达式浓缩到一句话中。
- 列表推导式：[表达式 + for语句 +(if语句)]
- 将满足条件的值用于计算表达式后得到的值加入到列表中
- **In [149]: strings = ['a', 'as', 'bat', 'car', 'dove', 'python']**
- **...: [x.upper() for x in strings if len(x) > 2]**
- **Out[149]: ['BAT', 'CAR', 'DOVE', 'PYTHON']**

- **In [150]: unique_lengths = {len(x) for x in strings}**
- **...: unique_lengths**
- **Out[150]: {1, 2, 3, 4, 6}**



- 输入输出
- 数据类型
- 控制结构
- 函数定义
- 异常处理
- 面向对象

函数定义

Index

函数定义

- Python的函数定义需要“def”关键字，一个函数名，一组参数，一个函数体，并可能显式返回值
- def function(para1, para2, …):

  …

  return …
- 一旦return，则跳出该函数体，后面的语句将不再执行

函数定义

- In [151]: def my_function(x, y, z=1.5):
- ...: if z > 1:
- ...: return z * (x + y)
- ...: else:
- ...: return z / (x + y)
- ...:

- In [152]: my_function(5, 6, z=0.7)
- Out[152]: 0.06363636363636363

- In [153]: my_function(3.14, 7, 3.5)
- Out[153]: 35.49

参数类型

- 大家是否见过func(*args, **kwargs)这样的函数定义，这个*和**让人有点费解。
- 为了方便大家今后查询手册，查询文档，需要介绍python中函数参数的类型。
- 首先，可选参数
- def funcC(a, b=0):

```
    print a
    print b
```

参数类型

- 其次，可变数量参数：
- **In [158]: def funcD(a, b, *c):**
- **...: print (a)**
- **...: print (b)**
- **...: print ("length of c is: %d " % len(c))**
- **...: print (c)**

- **In [159]: funcD(1, 2, 3, 4, 5, 6)**
- **1**
- **2**
- **length of c is: 4**
- **(3, 4, 5, 6)**

参数类型

- 再次，位置参数和关键字参数：
- **In [161]: def funcE(a, b, c):**
- **...: print (a)**
- **...: print (b)**
- **...: print (c)**

- **In [162]: funcE(100, 99, 98)**
- **100**
- **99**
- **98**

- **In [163]: funcE(100, c=98, b=99)**
- **100**
- **99**
- **98**

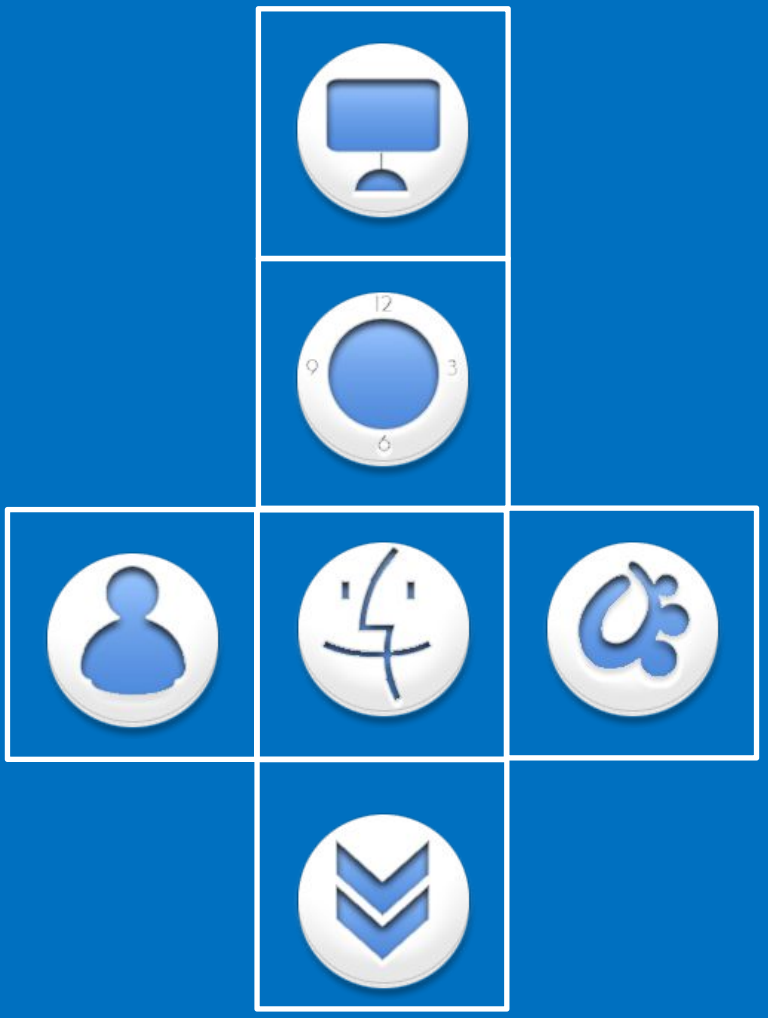
参数类型

- 最后，**的含义
- **In [165]: def funcF(a, **b):**
- **...: print (a)**
- **...: for x in b:**
- **...: print (x + ": " + str(b[x]))**

- **In [166]: funcF(100, c='hello', b=200)**
- **100**
- **c: hello**
- **b: 200**

扩展阅读

- 你可能永远不会用到下面的东西，但是不妨碍你了解他们。
- Lambda函数，闭包，currying，生成器
- 生成器，itertools



- 输入输出
- 数据类型
- 控制结构
- 函数定义
- 异常处理
- 面向对象

异常处理

Index

- 异常类型
- 异常处理
- 自建异常

异常类型

- 1、NameError：尝试访问一个未申明的变量
  >>> v

  2、ZeroDivisionError：除数为0
  >>> v = 1/0

  3、SyntaxError：语法错误
  >>> int int

  4、IndexError：索引超出范围
  >>> List = [2]
  >>> List[3]

  5、KeyError：字典关键字不存在
  >>> Dic = {'1':'yes', '2':'no'}
  >>> Dic['3']

异常类型

- 6、IOError：输入输出错误
  >>> f = open('abc')

- 7、AttributeError：访问未知对象属性
  >>> dict1.reverse()

- 8、ValueError：数值错误
  >>> int(‘d’)
  ValueError: invalid literal for int() with base 10: ‘d’

- 9、TypeError：类型错误
  >>> iStr = ’22’　　>>> iVal = 22
  >>> obj = iStr + iVal;

- 10、AssertionError：断言错误
  >>> assert 1 != 1

异常处理

- 使用try-except语句，我们可以对异常进行处理
- 语句结构：
- try:

```
        # 可能产生异常的语句
    except:
        # 当异常发生时执行的语句
    else:
        #try成功时候执行
    finally:
        #无论如何都执行
```

异常处理

- f = open(path, 'w')

- try:
- write_to_file(f)
- except:
- print 'Failed'
- else:
- print 'Succeeded'
- finally:
- f.close()

自建异常

- 可以使用raise语句自己抛出异常
- class ResultError(IOError): pass

- if score>90:
- raise ResultError('Can\'t be that good!')
- else:
- print(score)
- ---
- ResultError Traceback (most recent call last)
- <ipython-input-173-951814d5b258> in <module>()
- 1 if score>90:
- ----> 2 raise ResultError('Can\'t be that good!')
- 3 else:
- 4 print(score)

- ResultError: Can't be that good!

DS&A

Thank You

designed by Sun