Python语言项目教学02

陈斌

北京大学地球与空间科学学院

2018.10.26

目录

- Python语言要件概览
- 数值和逻辑类型
- 字符串类型
- 上机练习
- 容器类型
- 大型组合数据结构
- 上机练习

北京大学 陈斌 gischen@pku.edu.cn

Python语言的几个要件

数据对象和组织

- 对现实世界实体和概念的抽象
- 分为简单类型和容器类型
- 简单类型用来表示值
  - 整数int、浮点数float、复数complex、逻辑值bool、字符串str
- 容器类型用来组织这些值
  - 列表list、元组tuple、集合set、字典dict
- 数据类型之间几乎都可以转换

赋值和控制流

- 对现实世界处理和过程的抽象
- 分为运算语句和控制流语句
- 运算语句用来实现处理与暂存
  - 表达式计算、函数调用、赋值
- 控制流语句用来组织语句描述过程
  - 顺序、条件分支、循环
- 定义语句也用来组织语句，描述一个包含一系列处理过程的计算单元
  - 函数定义、类定义

Python数据类型：整数int、浮点数float

- 最大的特点是不限制大小
- 浮点数受到17位有效数字限制
- 常见的运算包括加、减、乘、除、整除、求余、幂指数等
- 浮点数的操作也差不多
- 一些常用的数学函数如sqrt/sin/cos等都在math模块中
  - `import math`
  - `math.sqrt(2)`

```
>>> 5
5
>>> -100
-100
>>> 5 + 8
13
>>> 90 - 10
80
>>> 4 * 7
28
>>> 7 / 2
3.5
>>> 7 // 2
3
>>> 7 % 3
1
>>> 3 ** 4
81
>>> 2 ** 100
1267650600228229401496703205376
>>> divmod(9, 5)
(1, 4)
>>>
```

数值常见的运算和比较

运算符	功能	备注
`m + n`	加法	
`m - n`	减法	
`m * n`	乘法	
`m // n`	整数除法	结果是商的整数部分
`m / n`	除法	“真”除法，得到小数
`m % n`	求余数	
`divmod(m, n)`	求整数除法和余数	会得到两个整数，一个是 `m // n`，另一个是 `m % n`
`m ** n`	求乘方	整数 `m` 的 `n` 次方
`abs(m)`	求绝对值	
`m == n`	相等比较	`m` 是否等于 `n`
`m > n`	大于比较	`m` 是否大于 `n`
`m >= n`	大于或等于比较	`m` 是否大于或者等于 `n`
`m < n`	小于比较	`m` 是否小于 `n`
`m <= n`	小于或等于比较	`m` 是否小于或者等于 `n`

- 可以进行连续比较判断
  - `>>> 7 > 3 >= 3`
  - `True`
  - `>>> 12 < 23 < 22`
  - `False`

整数的进制

进制	表示	例子
十进制decimal	无前缀数字	367
二进制binary	0b前缀	0b101101111
八进制octal	0o前缀	0o557
十六进制 hexadecimal	0x前缀	0x16f

- 可以用各种进制表示整数
- 也可以转为字符串
  - `str(), bin(), oct(), hex()`
- 浮点数可以转为十六进制
  - `float.hex()`

```
>>> float.hex(1.23)
'0x1.3ae147ae147aep+0'
>>> (1.23).hex()
'0x1.3ae147ae147aep+0'
```

浮点数的精度问题

- 计算机内部用二进制保存数值，
- 十进制的有限小数转为二进制可能变成无限循环小数
  - $(0.1)_{10}=(0.000110011001\ldots)_2$
- 四舍五入将产生误差
- 浮点数判断相等不能简单用相等关系符判断
- 可以视数值取小数点后固定位数进行四舍五入再判断相等

```
>>> 0.2+0.1
0.30000000000000004
>>> 0.2+0.1==0.3
False
>>> round(0.2+0.1, 10)==round(0.3, 10)
True
>>>
```

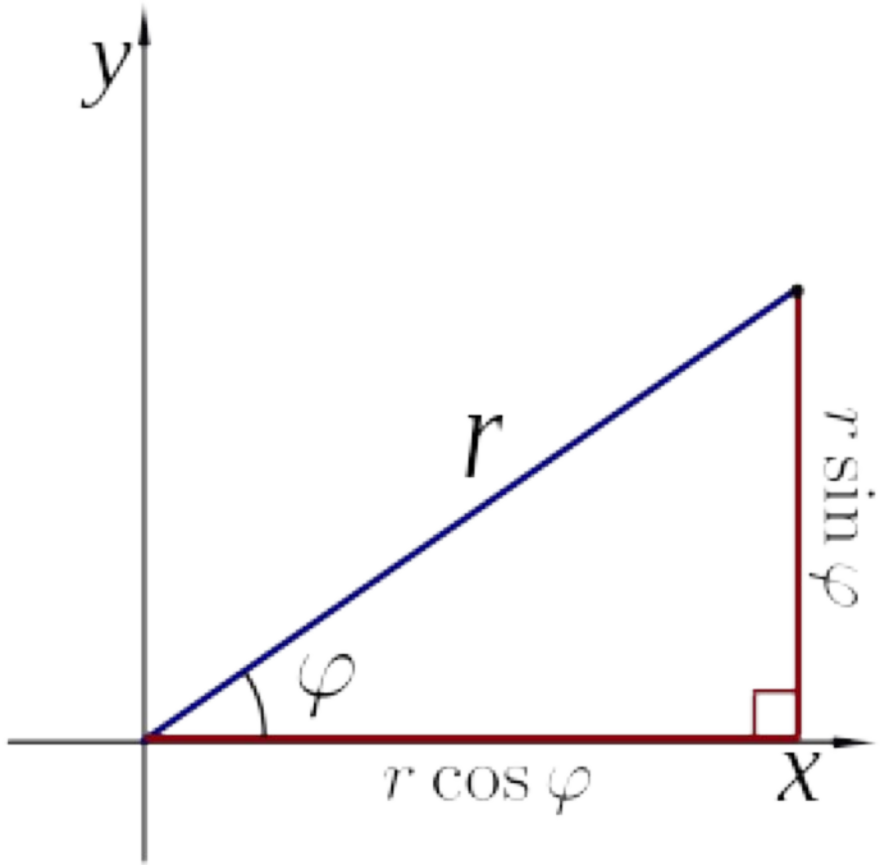
Python数据类型：复数

- Python内置复数类型
  - `<class 'complex'>`
- 支持所有常见的复数计算
  - abs函数支持复数取模运算
- 对复数处理的数学函数在模块cmath中
  - `import cmath`
  - `cmath.sqrt(1+2j)`

```
>>> 1+3j
(1+3j)
>>> (1+2j)*(2+3j)
(-4+7j)
>>> (1+2j)/(2+3j)
(0.6153846153846154+0.07692307692307691j)
>>> (1+2j)**2
(-3+4j)
>>> (1+2j).imag
2.0
>>> (1+2j).real
1.0
>>>
```

```
>>> abs(1+2j)
2.23606797749979
```

Python数据类型：复数的形式转换



polar：极坐标
rect：直角坐标

```
>>> import cmath
>>> cmath.polar(3+4j)
(5.0, 0.9272952180016122)
>>> cmath.rect(1, cmath.pi)
(-1+1.2246467991473532e-16j)
>>>
```

Python数据类型：逻辑值

- 逻辑值仅包括True/False两个
- 用来配合if/while等语句做条件判断
- 其它数据类型可以转换为逻辑值：
  - 数值：0与非0
  - 字符串：空串与非空串
  - 容器：空容器与非空容器
  - None是False

```
>>> True
True
>>> False
False
>>> 1>2
False
>>> 23<=34
True
>>> bool(0)
False
>>> bool(999)
True
>>> if (2>1):
        print ("OK")


OK
>>>
```

Python数据类型：字符串

- 最大的特点是Python字符串不可修改，只能生成新的字符串
- 用双引号或者单引号都可以表示字符串
- 多行字符串用三个连续引号表示
- 特殊字符用转义符号“\”表示
  - 制表符\t，换行符号\n
- 字符串操作：
  - +连接、*复制、len长度
  - [start:end:step]用来提取一部分

```
>>> 'abc'
'abc'
>>> "abc"
'abc'
>>> '''abc def
ghi jk'''
'abc def\nghi jk'
>>> "Hello\nWorld!"
'Hello\nWorld!'
>>> print ("Hello\nWorld!")
Hello
World!
>>> 'abc' + 'def'
'abcdef'
>>> 'abc' * 4
'abcabcabcabc'
>>> len('abc')
3
>>> 'abcd'[0:2]
'ab'
>>> 'abcd'[0::2]
'ac'
```

字符串str和字节串bytes

- Python语言中的字符串是unicode字符的串
- 可以通过encode()方法转换为各种字符编码的字节串bytes
  - 指定字符编码如gb2312，gbk等
- 而字节串则可以通过decode()方法转换为字符串str
  - 字节串属于特定字符编码

```
>>> type('中文')
<class 'str'>
>>> '中文'.encode()
b'\xe4\xb8\xad\xe6\x96\x87'
>>> '中文'.encode('gb2312')
b'\xd6\xd0\xce\xc4'
>>> type(b'\xd6\xd0')
<class 'bytes'>
>>> b'\xd6\xd0'.decode('gb2312')
'中'
>>>
```

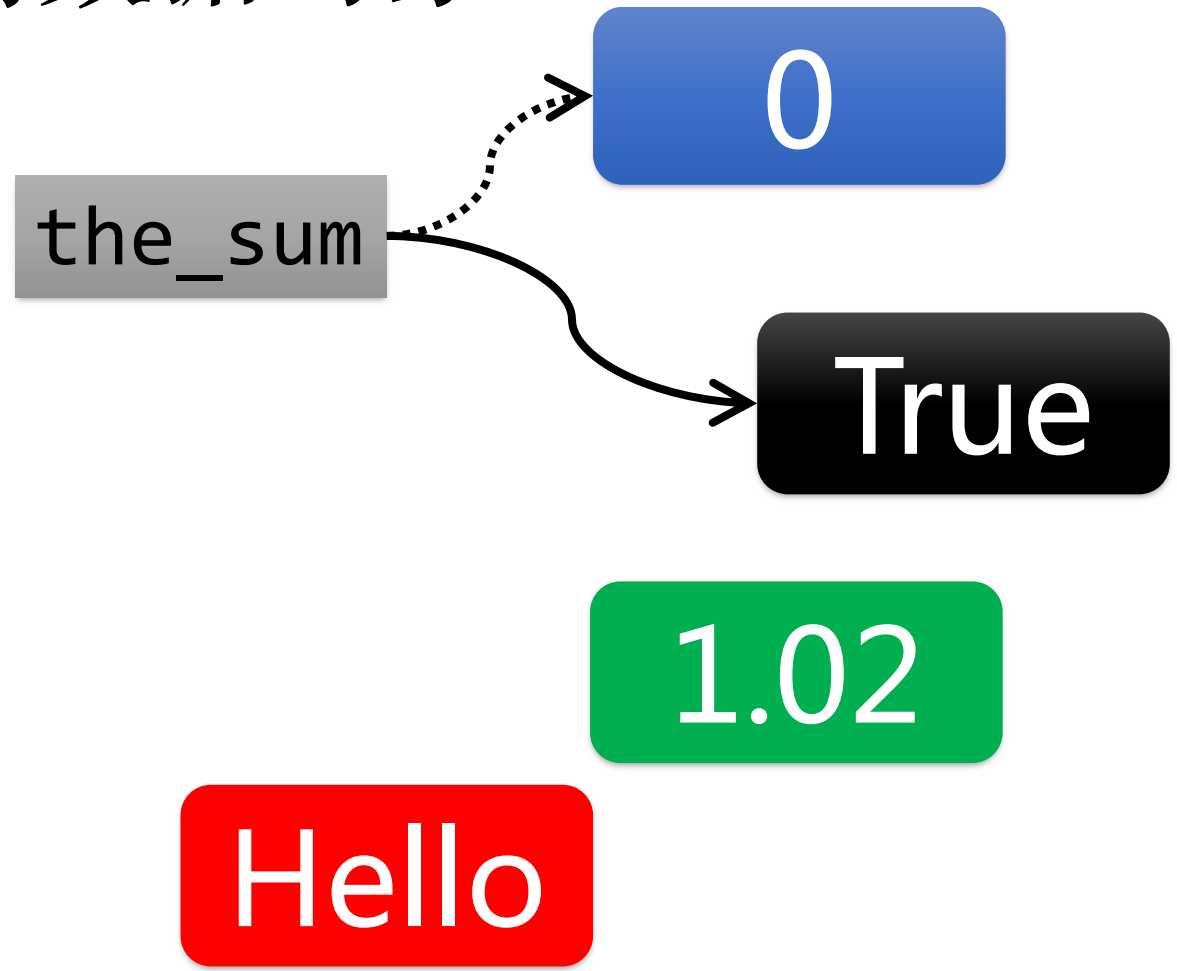
Python数据类型：字符串

- 一些高级操作：
  - split：分割；join：合并
  - upper/lower/swapcase：大小写相关
  - ljust/center/rjust：排版左中右对齐
  - replace：替换子串

```
>>> 'You are my sunshine.'.split(' ')
['You', 'are', 'my', 'sunshine.']
>>> '-'.join(["One", "for", "Two"])
'One-for-Two'
>>> 'abc'.upper()
'ABC'
>>> 'aBC'.lower()
'abc'
>>> 'Abc'.swapcase()
'aBC'
>>> 'Hello World!'.center(20)
'    Hello World!    '
>>> 'Tom smiled, Tom cried, Tom shouted'.replace('Tom', 'Jane')
'Jane smiled, Jane cried, Jane shouted'
```

Python变量机制：引用数据对象

- 赋值语句`the_sum = 0`，实际上是创建了名为the_sum的变量，然后指向数据对象“0”
- 所以变量可以随时指向任何一个数据对象，比如True，1.02，或者"Hello"
- 变量的类型随着指向的数据对象类型改变而改变！



上机练习：基本数据类型

- 数值基本运算：33和7
  - +, -, *, /, //, %, **
  - hex(), oct(), bin()
- 类型转换
  - 1, 0, 'abc', None, 1.2, False, ''
  - str(), bool(), int(), float()
  - is None, ==, !=
- 字符串基本操作
  - +, *, len(), [], in
  - ord(), chr()
  - 含有中文的字符串
- 字符串高级操作
  - s='abcdefg12345'
    - 切片：获得defg12，获得fg12345，获得54321， 获得aceg2
  - t='Mike and Tom'
    - split拆分、
    - upper/lower/swapcase修改大小写、
    - ljust/center/rjust排版30位宽度左中右对齐
    - replace将Mike替换为Jerry

Python容器类型：列表和元组

- Python中有几种类型是一系列元素组成的序列，以整数作为索引
- 字符串str是一种同类元素的序列
- 列表list和元组tuple则可以容纳不同类型的元素，构成序列
- 元组是不可更新（不可变）序列
  - 字符串也是不能再更新的序列
- 列表则可以删除、添加、替换、重排序列中的元素
  - 可变类型

字符串str

0	1	2	3	4	5	6	7	8
H	*e*	*l*	*l*	*o*		*T*	*o*	*m*

列表list

0	1	2	3	4	5	6	7	8
123	2.4	'ab'	True	None	[1,2]	(2,3)	556	0

元组tuple

0	1	2	3	4	5	6	7	8
123	2.4	'ab'	True	None	[1,2]	(2,3)	556	0

容器类型：列表和元组

- 创建列表：[ ]或者list()
- 创建元组：( )或者tuple()
- 用索引[n]获取元素（列表可变）
- +：连接两个列表/元组
- *：复制n次，生成新列表/元组
- len()：列表/元组中元素的个数
- in：某个元素是否存在
- [start : end : step]：切片

```
>>> []
[]
>>> list()
[]
>>> alist = [1, True, 0.23
>>> alist[0]
1
>>> alist + ["Hello"]
[1, True, 0.234, 'Hello']
>>> alist * 2
[1, True, 0.234, 1, True,
>>> len(alist)
3
>>> 1 in alist
True
>>> alist
[1, True, 0.234]
>>> alist[1:3]
[True, 0.234]
>>> alist[0:3:2]
[1, 0.234]
>>> alist[::-1]
[0.234, True, 1]
```

```
>>> ()
()
>>> tuple()
()
>>> atuple = (1, True, 0.234)
>>> atuple[0]
1
>>> atuple + ("Hello",)
(1, True, 0.234, 'Hello')
>>> atuple * 2
(1, True, 0.234, 1, True, 0.234)
>>> len(atuple)
3
>>> 1 in atuple
True
>>> atuple
(1, True, 0.234)
>>> atuple[1:3]
(True, 0.234)
>>> atuple[0:3:2]
(1, 0.234)
>>> atuple[::-1]
(0.234, True, 1)
```

```
>>> alist[0] = False
>>> alist
[False, True, 0.234]
```

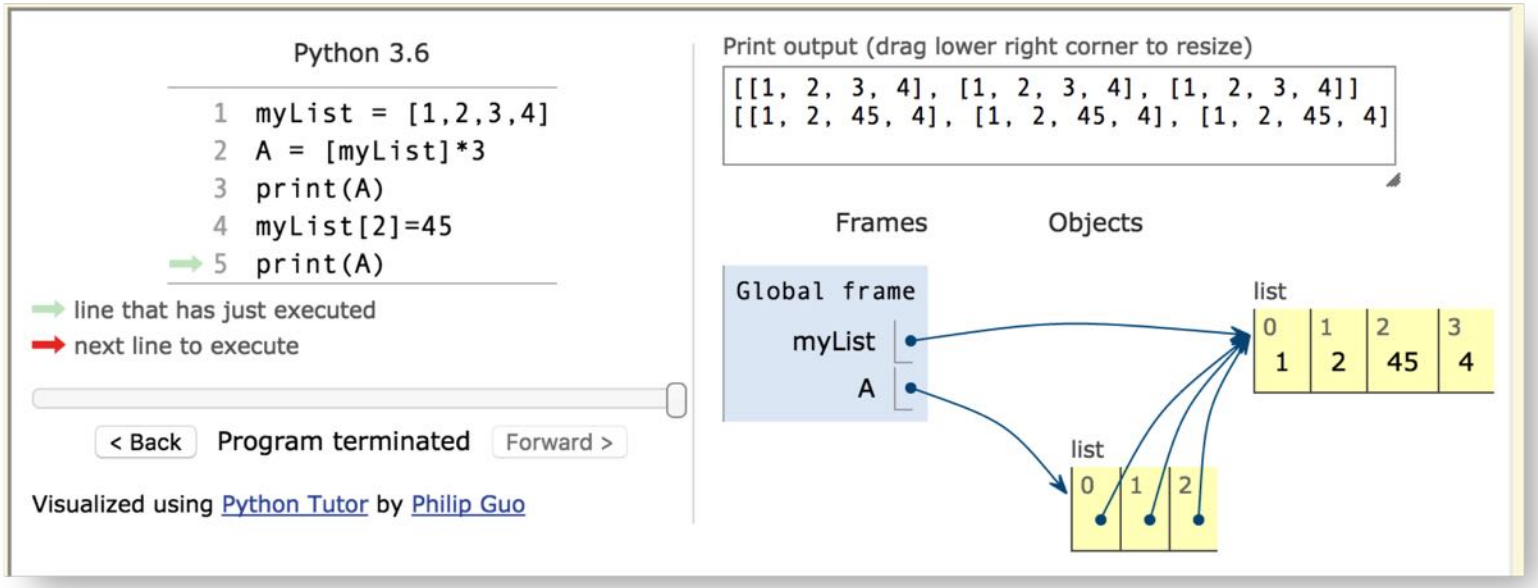
```
>>> atuple[0] = False
Traceback (most recent call last):
  File "<pyshell#93>", line 1, in <module>
    atuple[0] = False
TypeError: 'tuple' object does not support item assignment
```

列表list的其它方法

方法名称	使用例子	说明
append	alist.append(item)	列表末尾添加元素
insert	alist.insert(i,item)	列表中i位置插入元素
pop	alist.pop()	删除最后一个元素，并返回其值
pop	alist.pop(i)	删除第i个元素，并返回其值
sort	alist.sort()	将表中元素排序
reverse	alist.reverse()	将表中元素反向排列
del	del alist[i]	删除第i个元素
index	alist.index(item)	找到item的首次出现位置
count	alist.count(item)	返回item在列表中出现的次数
remove	alist.remove(item)	将item的首次出现删除

可变类型的变量引用情况

- 由于变量的引用特性，可变类型的变量操作需要注意
- 多个变量通过赋值引用同一个可变类型对象时
- 通过其中任何一个变量改变了可变类型对象
- 其它变量也看到了改变
  - alist = [1,2,3,4]
  - blist = alist
  - blist[0] = 'abc'
  - clist = alist[:]
  - Clist[0] = None



常用的连续序列生成器：range函数

- range(n)
  - 从0到n-1的序列
- range(start, end)
  - 从start到end-1的序列
- range(start, end, step)
  - 从start到end-1，步长间隔step
  - step可以是负数
- range函数返回range类型的对象，可以直接当做序列用，也可以转换为list或者tuple等容器类型

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(5, 10))
[5, 6, 7, 8, 9]
>>> list(range(1, 10, 2))
[1, 3, 5, 7, 9]
>>> list(range(10, 1, -2))
[10, 8, 6, 4, 2]
>>> range(10)
range(0, 10)
>>> tuple(range(10))
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Python容器类型：集合set

- 集合是不重复元素的无序组合
- 用set()从其它序列转换生成集合
- 集合的常见操作
  - in: 判断元素是否属于集合
  - |, union(): 并集
  - &, intersection() : 交集
  - -, difference() : 差集
  - ^, symmetric_difference() : 异或
  - <=,<,>=,>: 子集/真子集/超集/真超集

```
>>> set()
set()
>>> aset = set('abc')
>>> aset
{'c', 'a', 'b'}
>>> 'a' in aset
True
>>> aset | set('bcd')
{'c', 'd', 'a', 'b'}
>>> aset & set(['b', 'c', 'd'])
{'c', 'b'}
>>> aset - set(('b', 'c', 'd'))
{'a'}
>>> aset ^ set('bcd')
{'a', 'd'}
>>> aset <= set('abcd')
True
>>> aset > set('abcd')
False
```

Python容器类型：集合set

- add(x)：集合中添加元素
- remove(x)：删除指定元素
- pop()：删除集合中任意元素并返回其值
- clear()：清空集合成为空集
- 如果经常需要判断元素是否在一组数据中，这些数据的次序不重要的话，推荐使用集合，可以获得比列表更好的性能

```
>>> aset
{'c', 'a', 'b'}
>>> aset.add(1.23)
>>> aset
{'c', 1.23, 'a', 'b'}
>>> aset.remove('b')
>>> aset
{'c', 1.23, 'a'}
>>> aset.pop()
'c'
>>> aset
{1.23, 'a'}
>>> aset.clear()
>>> aset
set()
```

Python容器类型：字典dict

- 字典是通过键值key来索引元素value，而不是象列表是通过连续的整数来索引
- 字典是可变类型，可以添加、删除、替换元素
- 字典中的元素value没有顺序，可以是任意类型
- 字典中的键值key须是不可变类型（数值/字符串/元组）

```
>>> student = {'name':'Tom', 'age':20,
               'gender':'Male', 'course':['math', 'computer']}
>>> student
{'name': 'Tom', 'age': 20, 'course': ['math', 'computer'], '
gender': 'Male'}
>>> student['name']
'Tom'
>>> student['age']
20
>>> student['age'] = 19
>>> student
{'name': 'Tom', 'age': 19, 'course': ['math', 'computer'], '
gender': 'Male'}
>>> student['course'].append('chemistry')
>>> student
{'name': 'Tom', 'age': 19, 'course': ['math', 'computer', 'c
hemistry'], 'gender': 'Male'}
>>> 'gender' in student
True
>>> student.keys()
dict_keys(['name', 'age', 'course', 'gender'])
>>> student.values()
dict_values(['Tom', 19, ['math', 'computer', 'chemistry'], '
Male'])
>>> student.items()
dict_items([('name', 'Tom'), ('age', 19), ('course', ['math'
, 'computer', 'chemistry']), ('gender', 'Male')])
```

建立大型数据结构

- 嵌套列表
  - 列表的元素是一些列表
  - alist[i][j]
- 字典的元素可以是任意类型，甚至也可以是字典
  - bands={'Marxes':['Moe','Curly']}
- 字典的键值可以是任意不可变类型，例如用元组来作为坐标，索引元素
  - poi={(100,100):'bus stop'}

```
>>> alist=[ [23, 34, 45], [True, 'ab']]
>>> alist[0][2]
45
>>> bands={'Marxes':['Moe','Curly'], 'KK':[True, 'moon']}
>>> bands['KK'][0]
True
>>> poi={(100,100):'Zhongguancun', (123,23):'Pizza'}
>>> poi[(100,100)]
'Zhongguancun'
```

输入和输出： input/print函数

- input(prompt)
  - 显示提示信息prompt，用户输入的内容以字符串形式返回
- print(v1, v2, v3, ...)
  - 打印各变量的值输出
  - 可以带参数end='\n'，缺省为换行，表示打印后以这个字符串结尾
  - 带参数sep=' '，缺省是空格，表示变量之间用什么字符串隔开
- 格式化字符串
  - '%d %s' % (v1, v2)

```
>>> yname = input ("Please input your name")
Please input your nameTom Hanks
>>> yname
'Tom Hanks'
>>> print (1, 23, 'Hello')
1 23 Hello
>>> print (1, 23, 'Hello', end='')
1 23 Hello
>>> print (1, 23, 'Hello', sep=',')
1,23,Hello
>>> '%d %s' % (23, 'Hello')
'23 Hello'
>>> '%d' % (23,)
'23'
>>> '(%4d):K:%s' % (12, 'Hello')
'(  12):K:Hello'
>>> '(%04d):K:%10s' % (12, 'Hello')
'(0012):K:     Hello'
```

上机练习

- 列表、元组基本操作
  - +, *, len(), [], in
- 列表、元组高级操作
  - mylist=[1,2,3,4,5]
  - 切片：获得[2,3,4]，获得[3,4,5]，获得[3,2,1]，获得[1,3,5]
  - mytpl=(1,2,3,4,5)同上操作
  - t='Mike and Tom'
  - split拆分、join合成为'Mike/and/Tom'
- 集合基本操作
  - a=set([1,2,3,4,5])
  - b=set([2,4,6,8,10])
  - 并、交、差、异或、子集
  - 添加、删除、是否空集
- 字典基本操作
  - mydict = { 1:'Mon', 'line1':3332 }
  - 添加、删除、是否空字典
  - 取字典所有的key / value
  - 判断key是否存在