Pyhton学习笔记

作者:Lee

认识python

python是一种解释性,面向对象型,动态数据库类型的高级编程语言

历史

1989年吉多.范罗苏姆开发出来python . 1991年开始发布

Python 能干啥

做网站, 爬虫, 数据分析, 人工智能, 测试, 运维　写微信公众号　写游戏
python很简单,简单到只需要引入第三方类库就可完成一些基本的功能.

Python的安装环境

python有两个版本: python2.x 和python3.x
作用: 运行python代码
1.安装python的时候,一定要将python添加到环境变量中
2.打开doc命令, 输入python,即可查看到python的版本内容信息
3.退出: exit()　quit()
4.pip 是python安装第三方工具的一个类库
　输入: pip -V　　#查看有没有pip这个库

安装编辑器

Pycharm
sublime

输入函数:

　　input() 返回值是一个字符串

Python算术运算符

Python算术运算符

以下假设变量a为10，变量b为21：

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 31
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -11
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 210
/	除 - x 除以 y	b / a 输出结果 2.1
%	取模 - 返回除法的余数	b % a 输出结果 1
**	幂 - 返回x的y次幂	a**b 为10的21次方
//	取整除 - 返回商的整数部分	9//2 输出结果 4 , 9.0//2.0 输出结果 4.0

比较运算符

Python比较运算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 True。
>	大于 - 返回x是否大于y	(a > b) 返回 False。
<	小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。	(a < b) 返回 True。
>=	大于等于 - 返回x是否大于等于y。	(a >= b) 返回 False。
<=	小于等于 - 返回x是否小于等于y。	(a <= b) 返回 True。

赋值运算符

Python赋值运算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

例子:
a = 2
c = 3
print(c%a)　结果为1
print(c//a)　结果为1 h

表格
升级 macOS 以查看此表格。

Python逻辑运算符

Python语言支持逻辑运算符，以下假设变量 a 为 10, b为 20:

运算符	逻辑表达式	描述	实例
and	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或" - 如果 x 是 True，它返回 x 的值，否则它返回 y 的计算值。	(a or b) 返回 10。
not	not x	布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b) 返回 False

与(and)　或(or)　非(not)
格式：表达式　and|or　表达式
总结：整个表达式也有两种结果,True ,False
逻辑与
表达式1　and　　表达式2　　整个表达式的值
真　　真　　真

真　　假　　假
假　　真　　假
假　　假　　假
总结: 只要有一个表达式为假,则整个表达式就为假

逻辑或:
表达式1　or　表达式2　整个表达式的值
真　　真　　真
真　　假　　真
假　　真　　真
假　　假　　假
总结:只要有一个表达式为真,则整个表达式的结果就为真.

逻辑非
not　表达式　整个表达式的值
真　　假
假　　真

成员运算符

Python成员运算符

除了以上的一些运算符之外，Python还支持成员运算符，测试实例中包含了一系列的成员，包括字符串，列表或元组。

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中 , 如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中 , 如果 x 不在 y 序列中返回 True。

身份运算符

Python身份运算符

身份运算符用于比较两个对象的存储单元

运算符	描述	实例
is	is 是判断两个标识符是不是引用自一个对象	**x is y**, 类似 **id(x) == id(y)** , 如果引用的是同一个对象则返回 True，否则返回 False
is not	is not 是判断两个标识符是不是引用自不同对象	**x is not y** ， 类似 **id(a) != id(b)**。如果引用的不是同一个对象则返回结果 True，否则返回 False。

注：**id() 函数**
用于获取对象内存地址。
id **语法**：id([object])
参数说明：
- object -- 对象。

返回值 :返回对象的内存地址。

***is* 与 == 区别：**

is 用于判断两个变量引用对象是否为同一个， == 用于判断引用变量的值是否相等。

运算符的优先级

运算符	描述
.	成员运算符
[] [:]	下标，切片
**	指数
~ + -	按位取反, 正负号
* / % //	乘，除，模，整除
+ -	加，减
>> <<	右移，左移
&	按位与
^ |	按位异或，按位或
<= < > >=	小于等于，小于，大于，大于等于
== !=	等于，不等于
is is not	身份运算符
in not in	成员运算符
not or and	逻辑运算符
= += -= *= /= %= //= **= &= `	=^=>>=<<=`

说明：在实际开发中，如果搞不清楚优先级可以使用括号来确保运算的执行顺序。

程序的短路原则:

表达式1 or 表达式2 .....表达式n
总结: 只要第一个确保表达式是为真,则后边的表达式不再进行判断,这样就大大的增加了程序运行的速度
表达式1 and 表达式2 .....表达式n
总结: 只要第一个确保表达式是为假,

python3 数字类型(Number)

Python 支持三种不同的数值类型：

- **整型(Int)** - 通常被称为是整型或整数，是正或负整数，不带小数点。Python3 整型是没有限制大小的，可以当作 Long 类型使用，所以 Python3 没有 Python2 的 Long 类型。
- **浮点型(float)** - 浮点型由整数部分与小数部分组成，浮点型也可以使用科学计数法表示（2.5e2 = 2.5 x 10^2 = 250）
- **复数((complex))** - 复数由实数部分和虚数部分构成，可以用a + bj,或者complex(a,b)表示， 复数的实部a和虚部b都是浮点型。

例如:

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j
080	32.3+e18	.876j
-0490	-90.	-.6545+0J
-0x260	-32.54e100	3e+26J
0x69	70.2-E12	4.53e-7j

- Python支持复数，复数由实数部分和虚数部分构成，可以用a + bj,或者complex(a,b)表示， 复数的实部a和虚部b都是浮点型。

Python 数字类型转换

有时候，我们需要对数据内置的类型进行转换，数据类型的转换，你只需要将数据类型作为函数名即可。

- **int(x)** 将x转换为一个整数。
- **float(x)** 将x转换到一个浮点数。
- **complex(x)** 将x转换到一个复数，实数部分为 x，虚数部分为 0。
- **complex(x, y)** 将 x 和 y 转换到一个复数，实数部分为 x，虚数部分为 y。x 和 y 是数字表达式。

例如:>>> a = 1.0
>>> int(a)

1

Python 数字运算

Python 解释器可以作为一个简单的计算器，您可以在解释器里输入一个表达式，它将输出表达式的值。
表达式的语法很直白： +, -, * 和 / 和其它语言（如Pascal或C）里一样。例如：

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # 总是返回一个浮点数
1.6
```

注意：在不同的机器上浮点运算的结果可能会不一样。
在整数除法中，除法（/）总是返回一个浮点数，如果只想得到整数的结果，丢弃可能的分数部分，可以使用运算符 **//**：

```
>>> 17 / 3  # 整数除法返回浮点型
5.666666666666667
>>>
>>> 17 // 3  # 整数除法返回向下取整后的结果
5
>>> 17 % 3  # %操作符返回除法的余数
2
>>> 5 * 3 + 2
17
```

变量在使用前必须先"定义"（即赋予变量一个值），否则会出现错误

数学函数:
max()
min()
pow()
round()

```
#求两个数的大小
a = 10
b = 15
print((a > b) - (a < b))
# 总结:如果a大于b,则返回1,如果a小于b,则返回-1,如果a等于b,则返回0
```

函数	返回值 (描述)
abs(x)	返回数字的绝对值，如abs(-10) 返回 10
ceil(x)	返回数字的上入整数，如math.ceil(4.1) 返回 5
cmp(x, y)	如果 x < y 返回 -1, 如果 x == y 返回 0, 如果 x > y 返回 1。 **Python 3 已废弃** 。使用 **使用 (x>y)-(x<y)** 替换。
exp(x)	返回e的x次幂(e^x),如math.exp(1) 返回2.718281828459045
fabs(x)	返回数字的绝对值，如math.fabs(-10) 返回10.0
floor(x)	返回数字的下舍整数，如math.floor(4.9)返回 4
log(x)	如math.log(math.e)返回1.0,math.log(100,10)返回2.0
log10(x)	返回以10为基数的x的对数，如math.log10(100)返回 2.0
max(x1, x2,...)	返回给定参数的最大值，参数可以为序列。
min(x1, x2,...)	返回给定参数的最小值，参数可以为序列。
modf(x)	返回x的整数部分与小数部分，两部分的数值符号与x相同，整数部分以浮点型表示。
pow(x, y)	x**y 运算后的值。
round(x [,n])	返回浮点数x的四舍五入值，如给出n值，则代表舍入到小数点后的位数。
sqrt(x)	返回数字x的平方根。

随机数函数

随机数可以用于数学，游戏，安全等领域中，还经常被嵌入到算法中，用以提高算法效率，并提高程序的安全性。

Python包含以下常用随机数函数：

函数	描述
choice(seq)	从序列的元素中随机挑选一个元素，比如random.choice(range(10))，从0到9中随机挑选一个整数。
randrange ([start,] stop [,step])	从指定范围内，按指定基数递增的集合中获取一个随机数，基数缺省值为1
random()	随机生成下一个实数，它在[0,1)范围内。
seed([x])	改变随机数生成器的种子seed。如果你不了解其原理，你不必特别去设定seed，Python会帮你选择seed。
shuffle(lst)	将序列的所有元素随机排序
uniform(x, y)	随机生成下一个实数，它在[x,y]范围内。

字符串:

字符串一般情况使用单引号或者双引号引起来
规则: 单不能套单,单可以套双
双不能套双,双可以套单

len():求字符串的长度

Python 访问字符串中的值

Python 不支持单字符类型，单字符在 Python 中也是作为一个字符串使用。
Python 访问子字符串，可以使用方括号来截取字符串

```
var1 = 'Hello World!'
var2 = “Helloword”

print ("var1[0]: ", var1[0])
print ("var2[1:5]: ", var2[1:5])
```

执行结果：

```
var1[0]:  H
var2[1:5]:  ello
```

字符串的运算:

下表实例变量a值为字符串 "Hello"，b变量值为 "Python"：

操作符	描述	实例
+	字符串连接	a + b 输出结果：HelloPython
*	重复输出字符串	a*2 输出结果：HelloHello
[]	通过索引获取字符串中字符	a[1] 输出结果 **e**
[:]	截取字符串中的一部分	a[1:4] 输出结果 **ell**
in	成员运算符 - 如果字符串中包含给定的字符返回 True	**'H' in a** 输出结果 1
not in	成员运算符 - 如果字符串中不包含给定的字符返回 True	**'M' not in a** 输出结果 1
r/R	原始字符串 - 原始字符串：所有的字符串都是直接按照字面的意思来使用，没有转义特殊或不能打印的字符。 原始字符串除在字符串的第一个引号前加上字母 `r`（可以大小写）以外，与普通字符串有着几乎完全相同的语法。	`print( r'\n' )` `print( R'\n' )`

字符串的截取:
'''

字符串截取:
str[0:3] #截取第一位到第三位的字符
str[:] #截取字符串的全部字符
str[6:] #截取第七个字符到结尾
str[:-3] #截取从头开始到倒数第三个字符之前
str[2] #截取第三个字符
str[-1] #截取倒数第一个字符
str[::-1] #创造一个与原字符串顺序相反的字符串
str[-3:-1] #截取倒数第三位与倒数第一位之前的字符
str[-3:] #截取倒数第三位到结尾
str1[2::-1] 从下标为4的值开始倒着取

```
var1 = 'Hello World!'


print(var1[4::-1])



结果:olleH

从下标为4的值开始倒着取
```

```
a = '0123456789'
print(a[:-5:-3])
倒着取,取到下标-5但不包含-5,-3是步长
结果是:96

print(a[-2:-8:-2])
倒着取,取到下标-8但不包含-8,-2是步长
结果为: 864
```

'''

字符串的格式化:

第一种格式化:
%s:　　给字符串站位
%d:　　给int类型站位
%f:　　给浮点类型站位，默认保留6为小数
　　%.2f: 保留两位小数
　　%10.2f 共10位,保留两位小数,其他为使用空格补齐
　　%010.2f 共10位,保留两位小数,其他为使用0补齐
%c:　　打印一个字符
%o:　　将十进制转换成八进制
%x:　　将十进制转换成十六进制

Python 的字符串内建函数

序号 方法及描述
1　　capitalize() 将字符串的第一个字符转换为大写
2　　center(width, fillchar)
返回一个指定的宽度 width 居中的字符串，fillchar 为填充的字符，默认为空格。
3　　count(str, beg= 0,end=len(string))
返回 str 在 string 里面出现的次数，如果 beg 或者 end 指定则返回指定范围内 str 出现的次数
4　　bytes.decode(encoding="utf-8", errors="strict")
Python3 中没有 decode 方法，但我们可以使用 bytes 对象的 decode() 方法来解码给定的 bytes 对象，这个 bytes 对象可以由 str.encode() 来编码返回。
5　　encode(encoding='UTF-8',errors='strict')
以 encoding 指定的编码格式编码字符串，如果出错默认报一个ValueError 的异常，除非 errors 指定的是'ignore'或者'replace'
6　　endswith(suffix, beg=0, end=len(string))
检查字符串是否以 obj 结束，如果beg 或者 end 指定则检查指定的范围内是否以 obj 结束，如果是，返回 True,否则返回 False.
7　　expandtabs(tabsize=8)
把字符串 string 中的 tab 符号转为空格，tab 符号默认的空格数是 8 。
8　　find(str, beg=0 end=len(string))
检测 str 是否包含在字符串中，如果指定范围 beg 和 end ，则检查是否包含在指定范围内，如果包含返回开始的索引值，否则返回-1
9　　index(str, beg=0, end=len(string))
跟find()方法一样，只不过如果str不在字符串中会报一个异常.
10　　isalnum()
如果字符串至少有一个字符并且所有字符都是字母或数字则返 回 True,否则返回 False
11　　isalpha()
如果字符串至少有一个字符并且所有字符都是字母则返回 True, 否则返回 False

12 isdigit()
如果字符串只包含数字则返回 True 否则返回 False..
13 islower()
如果字符串中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是小写，则返回 True，否则返回 False
14 isnumeric()
如果字符串中只包含数字字符，则返回 True，否则返回 False
15 isspace()
如果字符串中只包含空白，则返回 True，否则返回 False.
16 istitle()
如果字符串是标题化的(见 title())则返回 True，否则返回 False
17 isupper()
如果字符串中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是大写，则返回 True，否则返回 False
18 join(seq)
以指定字符串作为分隔符，将 seq 中所有的元素(的字符串表示)合并为一个新的字符串
19 len(string)
返回字符串长度
20 ljust(width[, fillchar])
返回一个原字符串左对齐,并使用 fillchar 填充至长度 width 的新字符串，fillchar 默认为空格。
21 lower()
转换字符串中所有大写字符为小写.
22 lstrip()
截掉字符串左边的空格或指定字符。
23 maketrans()
创建字符映射的转换表，对于接受两个参数的最简单的调用方式，第一个参数是字符串，表示需要转换的字符，第二个参数也是字符串表示转换的目标。
24 max(str)
返回字符串 str 中最大的字母。
25 min(str)
返回字符串 str 中最小的字母。
26 replace(old, new [, max])
把 将字符串中的 str1 替换成 str2,如果 max 指定，则替换不超过 max 次。
27 rfind(str, beg=0,end=len(string))
类似于 find()函数，不过是从右边开始查找.
28 rindex(str, beg=0, end=len(string))
类似于 index()，不过是从右边开始.
29 rjust(width,[, fillchar])
返回一个原字符串右对齐,并使用fillchar(默认空格）填充至长度 width 的新字符串
30 rstrip()
删除字符串字符串末尾的空格.
31 split(str="", num=string.count(str))
num=string.count(str)) 以 str 为分隔符截取字符串，如果 num 有指定值，则仅截取 num 个子字符串
32 splitlines([keepends])

按照行('\r', '\r\n', \n')分隔，返回一个包含各行作为元素的列表，如果参数 keepends 为 False，不包含换行符，如果为 True，则保留换行符。
33 startswith(str, beg=0,end=len(string))
检查字符串是否是以 obj 开头，是则返回 True，否则返回 False。如果beg 和 end 指定值，则在指定范围内检查。
34 strip([chars])
在字符串上执行 lstrip()和 rstrip()
35 swapcase()
将字符串中大写转换为小写，小写转换为大写
36 title()
返回"标题化"的字符串,就是说所有单词都是以大写开始，其余字母均为小写(见 istitle())
37 translate(table, deletechars="")
根据 str 给出的表(包含 256 个字符)转换 string 的字符, 要过滤掉的字符放到 deletechars 参数中
38 upper()
转换字符串中的小写字母为大写
39 zfill (width)
返回长度为 width 的字符串，原字符串右对齐，前面填充0
40 isdecimal()
检查字符串是否只包含十进制字符，如果是返回 true，否则返回 false。

Python3 列表

序列是Python中最基本的数据结构。序列中的每个元素都分配一个数字 - 它的位置，或索引，第一个索引是0，第二个索引是1，依此类推。
Python有6个序列的内置类型，但最常见的是列表和元组。
序列都可以进行的操作包括索引，切片，加，乘，检查成员。
此外，Python已经内置确定序列的长度以及确定最大和最小的元素的方法。
列表是最常用的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即可。如下所示：

```
list1 = ['Google', 'Runoob', 1997, 2000]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
list4 = [[1,2,3], 3,4,5]          #二维列表
```

与字符串的索引一样，列表索引从0开始。列表可以进行截取、组合等。

访问列表：

```
list1[0]      === ‘Google’
list4[0][1]   ===2
```

更新列表

你可以对列表的数据项进行修改或更新，你也可以使用append()方法来添加列表项

删除列表

使用del语句可以删除一个列表

del 列表名 可以直接删除一个列表

del list1[2] 删除列表中下标为2的元素

Python列表脚本操作符

列表对 + 和 * 的操作符与字符串相似。+ 号用于组合列表，* 号用于重复列表。

如下所示：

Python 表达式	结果	描述
len([1, 2, 3])	3	长度
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	组合
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	重复
3 in [1, 2, 3]	True	元素是否存在于列表中
for x in [1, 2, 3]: print(x, end=" ")	1 2 3	迭代

Python列表截取与拼接

Python的列表截取与字符串操类似，如下所示：

操作：

Python 表达式	结果	描述
L[2]	'Taobao'	读取第三个元素
L[-2]	'Runoob'	从右侧开始读取倒数第二个元素: count from the right
L[1:]	['Runoob', 'Taobao']	输出从第二个元素开始后的所有元素

列表函数的方法

序号	函数	
1	len(list)	列表元素个数
2	max(list)	返回列表元素最大值
3	min(list)	返回列表元素最小值

4	list(seq)	将元组转换为列表

序号	方法	
1	list.append(obj)	在列表末尾添加新的对象
2	list.count(obj)	统计某个元素在列表中出现的次数
3	list.extend(seq)	在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）
4	list.index(obj)	从列表中找出某个值第一个匹配项的索引位置
5	list.insert(index, obj)	将对象插入列表
6	list.pop(obj=list[-1])	移除列表中的一个元素（默认最后一个元素），并且返回该元素的值
7	list.remove(obj)	移除列表中某个值的第一个匹配项
8	list.reverse()	反向列表中元素
9	list.sort([func])	对原列表进行排序
10	list.clear()	清空列表
11	list.copy()	复制列表

Python3 元组

Python 的元组与列表类似，不同之处在于元组的元素不能修改。
元组使用小括号，列表使用方括号。
元组创建很简单，只需要在括号中添加元素，并使用逗号隔开即可。

创建元组:

1.创建一个空元组
tup1 = ()
2.创建只有一个元素的元组

```
a = '1',
b = (1,)
```

注意:元组中只包含一个元素时，需要在元素后面添加逗号，否则括号会被当作运算符使用：

```
>>> tup1 = (50)
>>> type(tup1)     # 不加逗号，类型为整型
<class 'int'>

>>> tup1 = (50,)
>>> type(tup1)     # 加上逗号，类型为元组
<class 'tuple'>
```

3.创建普通元组

```
tup1 = ('Google', 'Runoob', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"   #  不需要括号也可以
type(tup3)
<class 'tuple'>
```

4.创建二维元组

# 二维元祖:
tuple13 = ((2, 3, 4), (5, 6, 7))
# print(tuple13[1][1])

元组与字符串类似，下标索引从0开始，可以进行截取，组合等。

删除元组:

元组中的元素值是不允许删除的，但我们可以使用del语句来删除整个元组
示例如下:

```
tup = ('look', 'me', 2018, 2020)
print (tup)
del tup
print ("删除后的元组 tup : ")
print (tup)


Traceback (most recent call last):
('look', 'me', 2018, 2020)
  File "/Users/lizhonglin/Desktop/Code/test/cd.py", line 5, in <module>
    print (tup)
NameError: name 'tup' is not defined
```

访问元组:

元组可以使用下标索引来访问元组中的值，如下实例:

```
tuple4 = (2, 3, 4, 5, 6)
# print(tuple4[0])
# print(tuple4[1])
# 元祖在访问的时候一定不能溢出(越界),直接报错
# print(tuple4[5])
# 获取元祖中最后一个元素
# print(tuple4[-1])
# print(tuple4[-2])
# 一定不能越界
# print(tuple4[-6])
```

元组索引，截取

因为元组也是一个序列，所以我们可以访问元组中的指定位置的元素，也可以截取索引中的一段元素，如下所示：
元组：

```
tup1 = (1,2,3,4,5,6,7,8)
print(tup1[2:5])
print(tup1[2])
结果:
(3, 4, 5)
1
```

修改元组:

元组中的元素值是不允许修改的，但我们可以对元组进行连接组合

但是可以使用如下方法对元组进行操作

元组运算符

与字符串一样，元组之间可以使用 + 号和 * 号进行运算。这就意味着他们可以组合和复制，运算后会生成一个新的元组。

Python 表达式	结果	描述
len((1, 2, 3))	3	计算元素个数
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	连接
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	复制
3 in (1, 2, 3)	True	元素是否存在
for x in (1, 2, 3): print (x,)	1 2 3	迭代

元组内置函数

Python元组包含了以下内置函数

len()　　计算元组的元素个数

max()　　返回元组中元素的最大值

min()　　返回元组中元素的最小值

tuple(seq)　　将列表转换为元组

对元组的遍历:

```
# 对元祖进行遍历
# for x in (3, 4, 5, 6, 7):
#     print(x)

# range(): 从开始值开始,但不包含结束值[0, 10)
# for i in range(0, 10):
#     print(i)
```

Python3 字典

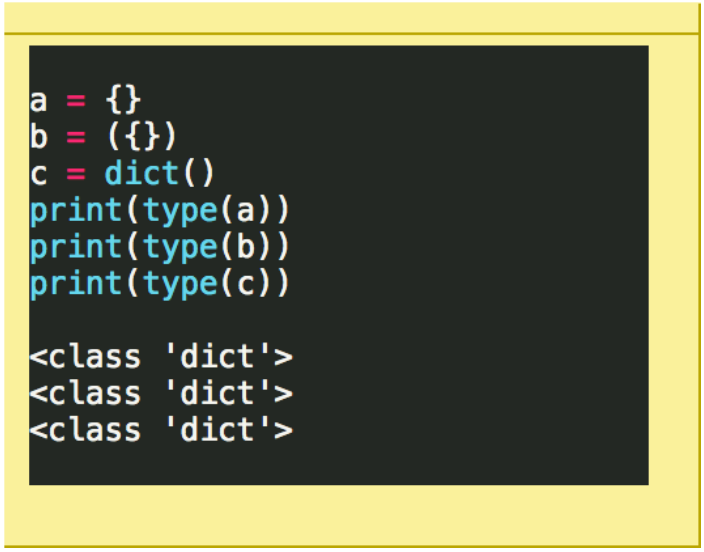
字典是另一种可变容器模型，且可存储任意类型对象。
字典的每个键值(key=>value)对用冒号(**:**)分割，每个对之间用逗号(**,**)分割，整个字典包括在花括号(**{}**)中 ,格式如下所示：

```
d = {key1 : value1, key2 : value2 }
```

键必须是唯一的，但值则不必。
值可以取任何数据类型，但键必须是不可变的，如字符串，数字或元组。
注意事项:
1.在字典中键(key)的值必须是唯一
2.在字典中可以存放多个键值对
3.在字典中键(key)必须是不可变类型　字符串　整数都可以作为键(key)
4.list和tuple都是有序集合, 而dictsahib无序集合

创建字典:

1.创建一个空字典



```
a = {}
b = ({})
c = dict()
print(type(a))
print(type(b))
print(type(c))

<class 'dict'>
<class 'dict'>
<class 'dict'>
```

2.普通的字典
dict1 = {'name' : '张三' , 'age' : 12}
dict2 = { 'abc': 123, 98.6: 37 }

访问字典里的值

访问字典的值:字典名[key]

修改字典

向字典添加新内容的方法是增加新的键/值对，修改或删除已有键/值对
例如:

```
dict1 = {'name' : '张三' , 'age' : 12}
dict1['name'] = '李四'
dict1['height'] = 178
print(dict1)

输出的结果:

{'name': '李四', 'age': 12, 'height': 178}
```

删除字典元素

能删单一的元素也能清空字典，清空只需一项操作。
显示删除一个字典用del命令

```
dict1 = {'name' : '张三' , 'age' : 12}
del dict1['name']  #删除键'name'
print(dict1)
结果为:{'age': 12}

dict1.clear()      #清空字典
print(dict1)
结果为:{}

del dict1          #删除字典
print(dict1)       #删除字典后在打印会引发异常如下

Traceback (most recent call last):
  File "/Users/lizhonglin/Desktop/Code/test/cd.py", line 8, in <module>
    print(dict1)     #删除字典后在打印会引发异常
NameError: name 'dict1' is not defined
```

字典键的特性

字典值可以是任何的 python 对象，既可以是标准的对象，也可以是用户定义的，但键不行。
两个重要的点需要记住：

1.不允许同一个键出现两次。创建时如果同一个键被赋值两次，后一个值会被记住

```
dict1 = {'name' : '张三' , 'age' : 12,'name':'王五'}

print(dict1)

结果:

{'name': '王五', 'age': 12}
```

2.键必须不可变，所以可以用数字，字符串或元组充当，而用列表就不行

```
#!/usr/bin/python3

dict = {['Name']: 'Runoob', 'Age': 7}

print ("dict['Name']: ", dict['Name'])
```

以上实例输出结果：

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Runoob', 'Age': 7}
TypeError: unhashable type: 'list'
```

字典内置函数&方法

Python字典包含了以下内置函数：

len(dict)	计算字典元素个数，即键的总数。
str(dict)	输出字典，以可打印的字符串表示。
type(variable)	返回输入的变量类型，如果变量是字典就返回字典类型。

Python字典包含了以下内置方法：

序号 函数及描述
1 radiansdict.clear() 删除字典内所有元素
2 radiansdict.copy() 返回一个字典的浅复制
3 radiansdict.fromkeys() 创建一个新字典，以序列seq中元素做字典的键，val为字典所有键对应的初始值
4 radiansdict.get(key, default=None) 返回指定键的值，如果值不在字典中返回default值
5 key in dict 如果键在字典dict里返回true，否则返回false
6 radiansdict.items() 以列表返回可遍历的(键, 值) 元组数组
7 radiansdict.keys() 以列表返回一个字典所有的键
8 radiansdict.setdefault(key, default=None) 和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default
9 radiansdict.update(dict2) 把字典dict2的键/值对更新到dict里
10 radiansdict.values() 以列表返回字典中的所有值
11 pop(key[,default]) 删除字典给定键 key 所对应的值，返回值为被删除的值。key值必须给出。 否则，返回default值。
12 popitem() 随机返回并删除字典中的一对键和值(一般删除末尾对)。

```
# 遍历
# for x in dict2:
#     print(x, dict2[x])
# 获取字典所有的value和key
# print(dict2.values())
# print(dict2.keys())

# for i in dict2.values():
#     print(i)
#
# for i in dict2.keys():
#     print(i)

# for k,v in dict2.items():
#     print(k, v)

for k,v in enumerate(dict2):
    print(k, v)
```

集合(set):

set(集合):类似于dict, 也是无序的,以key-value新的形势存在,但是没有value
作用: 是对list,tuple,dict进行去重的, 求交集.并集
1.set是无序

2.set集合是不可改变的

创建集合:

```
s = set()

print(type(s))
```

添加

```
set4 = set([3, 3, 4, 5, 7, 2, 1, 2])
set4.add(8)
# set4.add(3)#可以添加重复的值,但是没效果

# 总结:list和dict是可改变的, 而tuple是不可改变
# set4.add([10, 9])#直接报错,不能添加list
# set4.add((10, 9))
# set4.add({'a':1})#直接报错,不能添加字典
#
# print(set4)
```

修改

```
set5 = set([1, 2, 3, 4, 5])
# 将list dict tuple 等等整个插入进去
# set5.update([6, 7, 8])
# set5.update({9, 10})
# set5.update((11, 56))
# print(set5)
```

删除

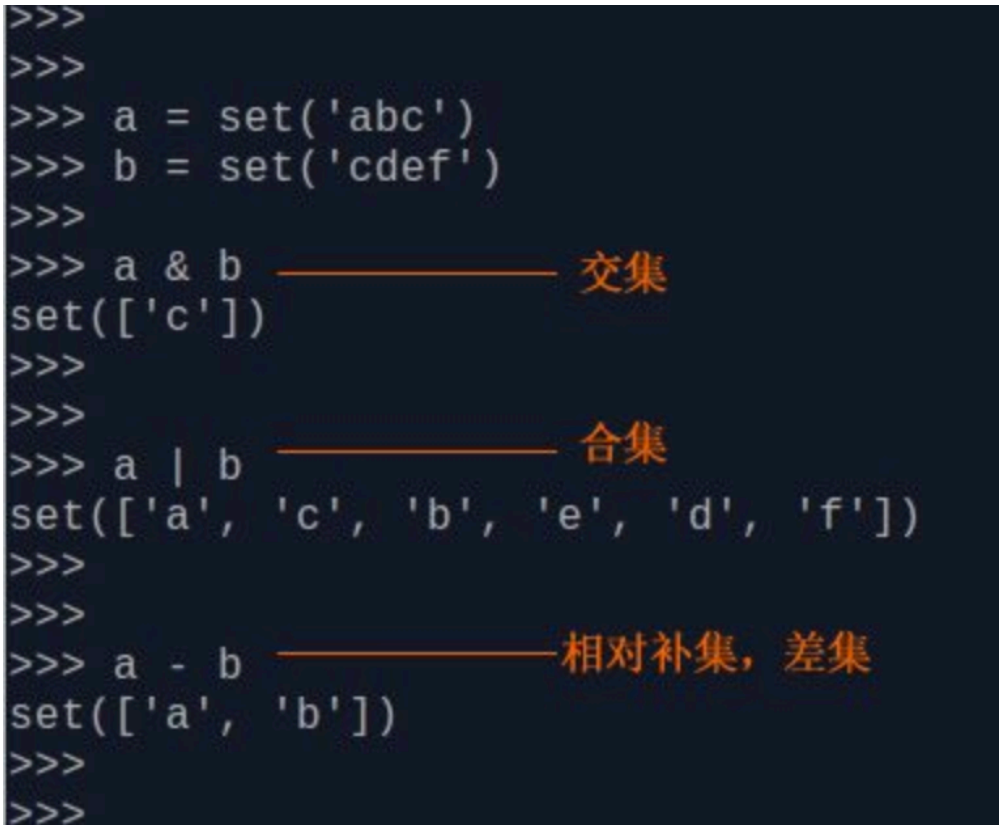
```
# set6= set([3, 4, 5, 6, 7])
# set6.remove(4)
# print(set6)
```

遍历

```
set7 = set([1, 2, 3, 4, 6])
set7 = set(['aaa', 'bbb', 'ccc'])
set7 = set((1, 2, 3, 2, 4, 2, 3))
# 在set集合中,没有value,即使有value也遍历不出来
set7 = set({'name':'小花', 'age': 18})
```

```
# for i in set7:
#     print(i, end = ',')
```

交集 差集 合集



列表、元组、集合、字典的区别

列表	元组	集合	字典	
英文	list	tuple	set	dict
可否读写	读写	只读	读写	读写
可否重复	是	是	否	是
存储方式	值	值	键(不能重复)	键值对(键不能重复)
是否有序	有序	有序	无序	无序，自动正序
初始化	`[1,'a']`	`('a', 1)`	`set([1,2])` 或 `{1,2}`	`{'a':1,'b':2}`
添加	`append`	只读	`add`	`d['key'] = 'value'`
读元素	`l[2:]`	`t[0]`	无	`d['a']`

列表、元组、集合、字典相互转换

列表元组转其他

```
1
2  # 列表转集合(去重)
3  list1 = [6, 7, 7, 8, 8, 9]
4  set(list1)
5  # {6, 7, 8, 9}
6
7  #两个列表转字典
8  list1 = ['key1','key2','key3']
9  list2 = ['1','2','3']
10 dict(zip(list1,list2))
11 # {'key1': '1', 'key2': '2', 'key3': '3'}
12
13 #嵌套列表转字典
14 list3 = [['key1','value1'],['key2','value2'],['key3','value3']]
15 dict(list3)
16 # {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
17
18 # 列表、元组转字符串
19 list2 = ['a', 'a', 'b']
20 ''.join(list2)
21 # 'aab'
22
23 tup1 = ('a', 'a', 'b')
24 ''.join(tup1)
25 # 'aab'
```

字典转其他

```
1 # 字典转换为字符串
2 dic1 = {'a':1,'b':2}
3 str(dic1)
4 # "{'a': 1, 'b': 2}"
5
6 # 字典key和value互转
7 dic2 = {'a': 1, 'b': 2, 'c': 3}
8 {value:key for key, value in a_dict.items()}
9 # {1: 'a', 2: 'b', 3: 'c'}
```

字符串转其他

```
# 字符串转列表
s = 'aabbcc'
list(s)
# ['a', 'a', 'b', 'b', 'c', 'c']

# 字符串转元组
tuple(s)
# ('a', 'a', 'b', 'b', 'c', 'c')

# 字符串转集合
set(s)
# {'a', 'b', 'c'}

# 字符串转字典
dic2 = eval("{'name':'ljq', 'age':24}")

# 切分字符串
a = 'a b c'
a.split(' ')
# ['a', 'b', 'c']
```

生成式::

Python3 函数

函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。
函数能提高应用的模块性，和代码的重复利用率。你已经知道Python提供了许多内建函数，比如print()。但你也可以自己创建函数，这被叫做用户自定义函数。

定义一个函数

你可以定义一个由自己想要功能的函数，以下是简单的规则：

- 函数代码块以 **def** 关键词开头，后接函数标识符名称和圆括号 ()。
- 任何传入参数和自变量必须放在圆括号中间，圆括号之间可以用于定义参数。
- 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
- 函数内容以冒号起始，并且缩进。
- **return [表达式]** 结束函数，选择性地返回一个值给调用方。不带表达式的return相当于返回 No

语法

Python 定义函数使用 def 关键字，一般格式如下：
def 函数名（参数列表）：
　　函数体
默认情况下，参数值和参数名称是按函数声明中定义的的顺序匹配起来的。

一个简单的函数:

```
def main():
    return 'hello world'
print(main())


结果:
hello world
```

函数调用

定义一个函数：给了函数一个名称，指定了函数里包含的参数，和代码块结构。
这个函数的基本结构完成以后，你可以通过另一个函数调用执行，也可以直接从 Python 命令提示符执行。
如下实例调用了 **printme()** 函数：

type() 函数

如果你只有第一个参数则返回对象的类型，三个参数返回新的类型对象。
isinstance() 与 *type*() 区别：

- *type*() 不会认为子类是一种父类类型，不考虑继承关系。

- *isinstance*() 会认为子类是一种父类类型，考虑继承关系。

如果要判断两个类型是否相同推荐使用 *isinstance*()。

语法

以下是 type() 方法的语法:
class type(name, bases, dict)

参数

- name -- 类的名称。
- bases -- 基类的元组。
- dict -- 字典，类内定义的命名空间变量。

返回值

一个参数返回对象类型, 三个参数，返回新的类型对象。