Python 内置函数

1、Python 内置函数简介

1、Python 内置函数 python 内置了一系列的常用函数，以便于我们使用。所有内置函数官网文档 https://docs.python.org/3/library/functions.html 内置函数：

		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

2、大家不要被这一堆内置函数吓到，有些常用的用多了就记得。不常用的记住 Python 有这个功能的函数，到时候回来看看资料就可以了。

2、数学运算

2.1、abs() 求绝对值,可以是整数，小数.

```
>>> abs(10)
10
>>> abs(-10)
10
>>> abs(-10.023)
10.023
>>>
```

2.2、round(2.6) 四舍五入取整，也就是 3.0

round(2.33365,3) 四舍五入保留几位小数

```
>>> round(2.6)
3
>>> round(2.4)
2
>>> round(-2.4)
-2
>>> round(2.465445,3)
2.465
>>>
```

2.3、pow(2, 3) 求指数，相当于 2**3

pow(2, 3, 5)，求指数之后再求余数，相当于 2**3 % 5

```
>>> pow(2,3)
8
>>> pow(2,3,2)
0
>>> pow(2,3,4)
0
>>> pow(2,3,5)
3
>>>
```

2.4、divmod(9,3) 返回除法结果和余数

```
>>> divmod(7,3)
(2, 1)
>>> divmod(9,3)
(3, 0)
>>>
```

2.5、max([1,5,2,9]) 返回最大值

```
>>> max(80,100,1000)
1000
>>> max(-20,-50,1,30)
30
>>> max(0,100,-300)
100
>>>
```

2.6、min([9,2,-4,2]) 返回最小值

```
>>> min(80, 100, 1000)
80
>>> min(-20, 100, 400)
```

```
-20
>>> min(-80, -20, -10)
-80
>>> min(0, 100, -400)
-400
>>>
```

2.7、sum([2,-1,9,12]) 求和

```
>>> max([1,5,2,6])
6
>>> min([1,5,2,6])
1
>>> sum([1,6,5,3])
15
```

2.8、eval(表达式，globals = None，locals = None)

参数是一个字符串和可选的全局变量和局部变量。如果提供， globals 必须是一个字典

```
>>> a=1
>>> b=2
>>> c =3
>>> eval('a+b')
3
>>> eval('a+b+c',{'c':3,'a':1,'b':3})
7
```

3、类型转换

3.1、int("5")转换为整数，将数字类型的字符串，浮点数，转换成整数

```
>>> int('666')
666
>>> int(3.6)
3
```

3.2、float(2) 转换为浮点数 float，将数字型的字符串，浮点数，转换成整数

```
>>> float(3)
3.0
>>> float('3')
3.0
>>>
```

3.3、str(2.3) 转换为字符串 string

```
>>> a = 1
>>> type(a)
```

```
<class 'int'>
>>> b = str(a)
>>> type(b)
<class 'str'>
>>>
```

3.4、ord("A") 转换 ASCII 字符为相应的数字

```
>>> ord('A')
65
>>> ord('a')
97
```

3.5、chr(65) 转换数字为相应 ASCII 码字符

```
>>> chr(65)
'A'
>>> chr(66)
'B'
>>> chr(67)
'C'
```

3.6、bool(0) 转换为相应的布尔值，在 Python 中，0 相当于 False，

下列对象都相当于 False： `[], (), {}, 0, None, 0.0, ''`

```
>>> bool(0)
False
>>> bool([])
False
>>> bool('a')
True
>>>
```

3.7、bin(32) 返回一个数字的二进制数。

```
>>> bin(52)
'0b110100'
>>> bin(12)
'0b1100'
```

3.8、hex(32) 返回一个数字的十六进制

```
>>> hex(32)
'0x20'
>>> hex(63)
'0x3f'
```

3.9、oct(56) 返回一个数字的八进制

```
>>> oct(56)
'0o70'
>>> oct(9)
'0o11'
>>> oct(8)
'0o10'
>>>
```

3.10、list((1,2,3)) 转换为表 list,可以将一个可迭代对象转成列表

```
>>> list((1,2,3))
[1, 2, 3]

>>>
```

3.11、tuple([2,3,4]) 转换为定值表 tuple

```
>>> tuple([2,3,4])
(2, 3, 4)
>>> tuple({'a':1,"b":2})
('b', 'a')
>>>
```

3.12、dict(a=1,b="hello",c=[1,2,3]) 构造字典 dictionary

```
>>> dict(a=1,b="hello",c=[1,2,3])
{'b': 'hello', 'a': 1, 'c': [1, 2, 3]}
>>>
```

3.13、bytes 返回一个字节对象

```
>>> bytes('中'.encode('utf-8'))
b'\xe4\xb8\xad'
>>> bytes('中'.encode('gbk'))
b'\xd6\xd0'
>>>
```

4、序列操作

4.1、all() 接受一个迭代器，如果迭代器的所有元素都为真，那么返回 True，否则返回 False

```
>>> li = [1,2,3]
>>> all(li)
True
>>> li = [1,2,3,0]
```

```
>>> all(li)
False
```

4.2、any() 接受一个迭代器，如果迭代器里有一个元素为真，那么返回 True,否则返回 False

```
>>> li = [1,2,3,0]
>>> any(li)
True
>>>
```

4.3、sorted 排序 sorted([1,555,4,6,8]，reverse=False) 也就是小到大排序。reverse=True 大到小排序

```
>>> sorted([1,2,3,5,6,9])
[1, 2, 3, 5, 6, 9]
>>> sorted([1,2,3,5,6,9],reverse=False)
[1, 2, 3, 5, 6, 9]
>>> sorted([1,2,3,5,6,9],reverse=True)
[9, 6, 5, 3, 2, 1]
>>> sorted(['a','b','C','D','d'],key=str.lower)  # 字符串无关大小写排序
['a', 'b', 'C', 'D', 'd']
```

4.4、reverse([1,5,3]) 返回一个倒序的序列

```
>>> a = [1,2,3,4,5]
>>> a.reverse()
>>> a
[5, 4, 3, 2, 1]
```

4.5、range() 生成序列。

range(stop) range(start, stop[, step]) start 开始，stop 结束位置，step 步长,生成一个 start，到 stop 的可迭代对象，左闭右开。

```
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>> for i in range(0,10,2):
...     print(i)
```

```
...
0
2
4
6
8
>>>
```

range 在 python2 中是直接生成列表，而在 python 中进行了优化不再直接生成列表，而是生成一个 range 可迭代对象。

4.6、zip(*iterables) 用于聚合来自每个迭代器的元素。

```
>>> zip([1,2,3],['a','b','c'])
<zip object at 0x7f5c2f9d7fc8>
>>>
>>> a = zip([1,2,3],['a','b','c'])
>>> list(a)
[(1, 'a'), (2, 'b'), (3, 'c')]
```

如果可迭代对象的元素个数不一样，那么按照最少的那个迭代压缩。最少元素那饿可迭代对象结束后退出。

```
>>> a = zip([1,2,3],['a','b','c','d','e'])
>>> list(a)
[(1, 'a'), (2, 'b'), (3, 'c')]
```

4.7、enumerate(iterable,start = 0)

返回一个枚举对象。iterable 必须是一个序列，一个 迭代器或者其他支持迭代的对象 enumerate()返回一个包含 count 的元组（从 start 开始，默认值为 0）以及 iterable 迭代获得的值。

```
>>> seasons =  ['spring','summer','fall','winter']
>>> list(enumerate(swasons))
[(0, 'spring'), (1, 'summer'), (2, 'fall'), (3, 'winter')]
>>> list(enumerate(swasons,start=5))
[(5, 'spring'), (6, 'summer'), (7, 'fall'), (8, 'winter')]
```

5、三元运算

5.1、什么是三元运算，听着就感觉高大上的感觉。

三元运算（三目运算），是对简单的条件语句的缩写，让代码更为简洁

5.2、三元运算格式

```
result = 值 1 if 条件 else 值 2
# 如果条件成立，那么将“值 1” 返回给 result 变量，否则将“值 2” 返回给 result 变量
```

普通 if-else 语句

```
# 普通 if-else
a = 1
b = 2

if a > b:
    print(a)
else:
    print(b)
```

利用三元运算改写

```
result = a if a > b else b
```

上面对比看，利用三元运算写简单的 if 语句会比一般 if-esle 简洁，阅读性更高。

6、Set 集合

6.1、set 也是 python 中的一种数据类型

set 集合，是一个无序且不重复的元素集合。

创建集合方式：

```
# 第一种方式
set1 = {"1","2"}
# 第二种方式
list1 = ['1','5','4','3']
set2 = set(list1)
```

6.2、集合常用方法

6.2.1、add 添加一个元素

```
>>> set1 = {'1','2'}
>>> set1.add('3')
>>> set1
{'2', '1', '3'}
>>>
```

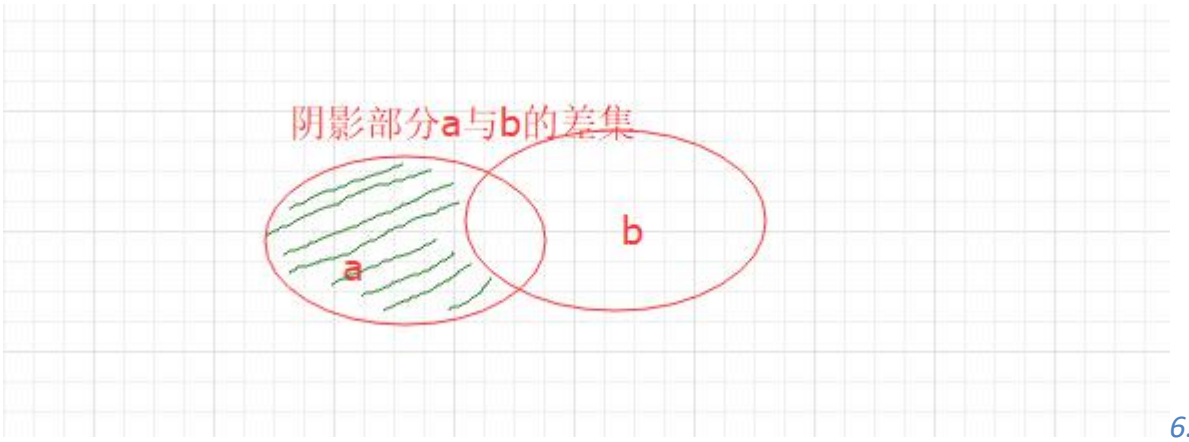
6.2.2、clear 清空集合的元素

```
>>> set1
{'2', '1', '3'}
>>>
>>> set1.clear()
>>> set1
set()
>>>
```

6.2.3、difference 两个集合的差集，a 中存在，b 中不存在

```
>>> a = {32,12,34}
>>> b = {12,43,23}
>>> a.difference(b)
{32, 34}
>>>
```

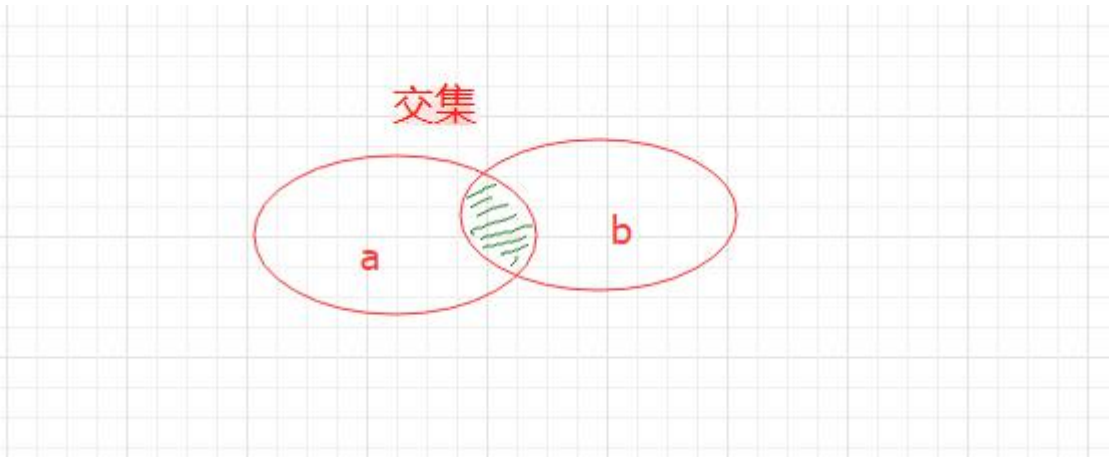
差集图示例：



6.2.4、intersection 两个集合的交集，a 中存在，b 中也存在的

```
>>> a = {32,12,34}
>>> b = {12,43,23}
>>> a.intersection(b)
{12}
>>>
```

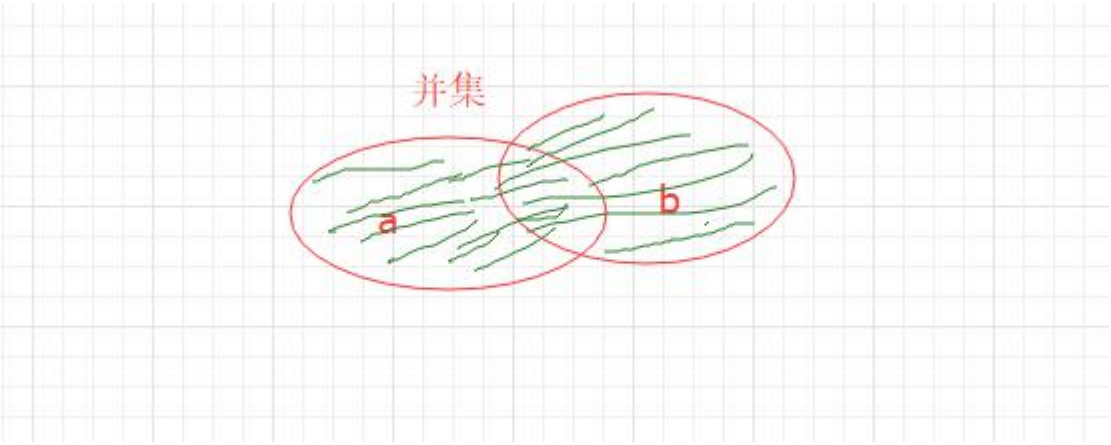
交集图示例



6.2.5、union 并集 包含 a 中的元素，也包含 b 中的元素

```
>>> a = {32,12,34}
>>> b = {12,43,23}
>>> a.union(b)
{32, 34, 23, 43, 12}
```

并集图示例



6.2.6、pop 集合 pop 随机移除某个元素并且获取那个参数,集合 pop 没有参数

```
>>> a =  {32,12,34}
>>> a.pop()
32
>>> a
{34, 12}
>>>
```

6.2.7、discard 移除指定元素

```
>>> a =  {32,12,34}
>>> a.discard(12)
>>> a
{32, 34}
>>>
```

6.2.8、update 更新集合

```
>>> a={1,2,3}
>>> b={4,5,6}
>>> a.update(b)
>>> a
{1, 2, 3, 4, 5, 6}
>>>
```