

Python流程控制与函数

本周内容

◆ 集合的操作

◆ 数据类型间的转换

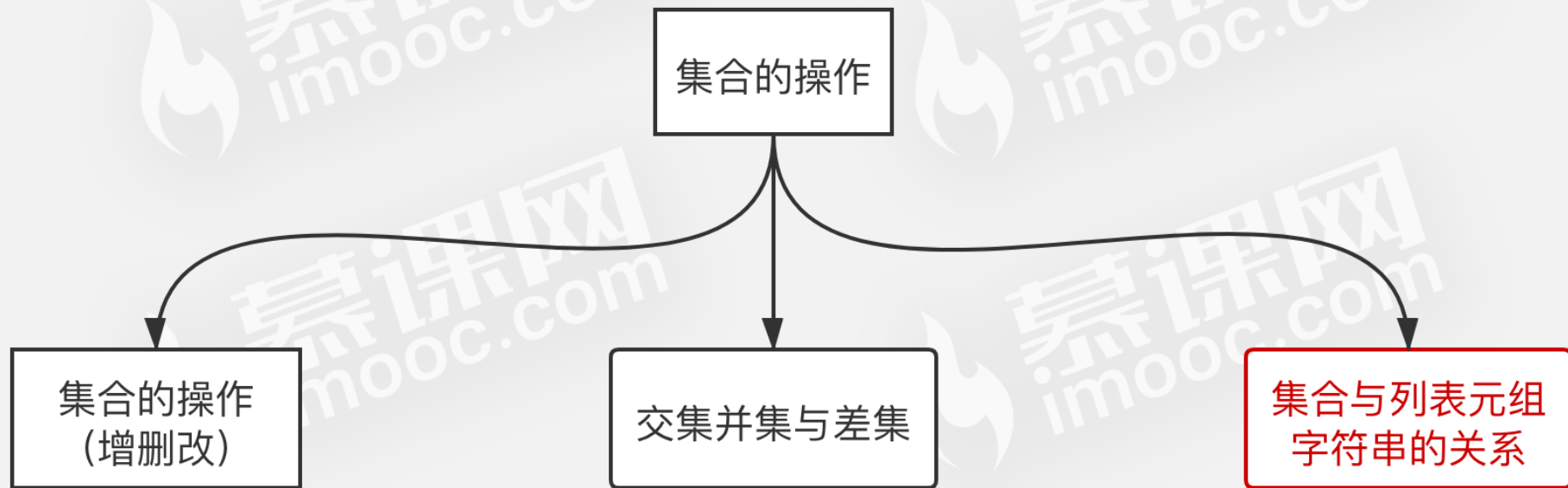
◆ 条件语句

◆ 循环

◆ 函数

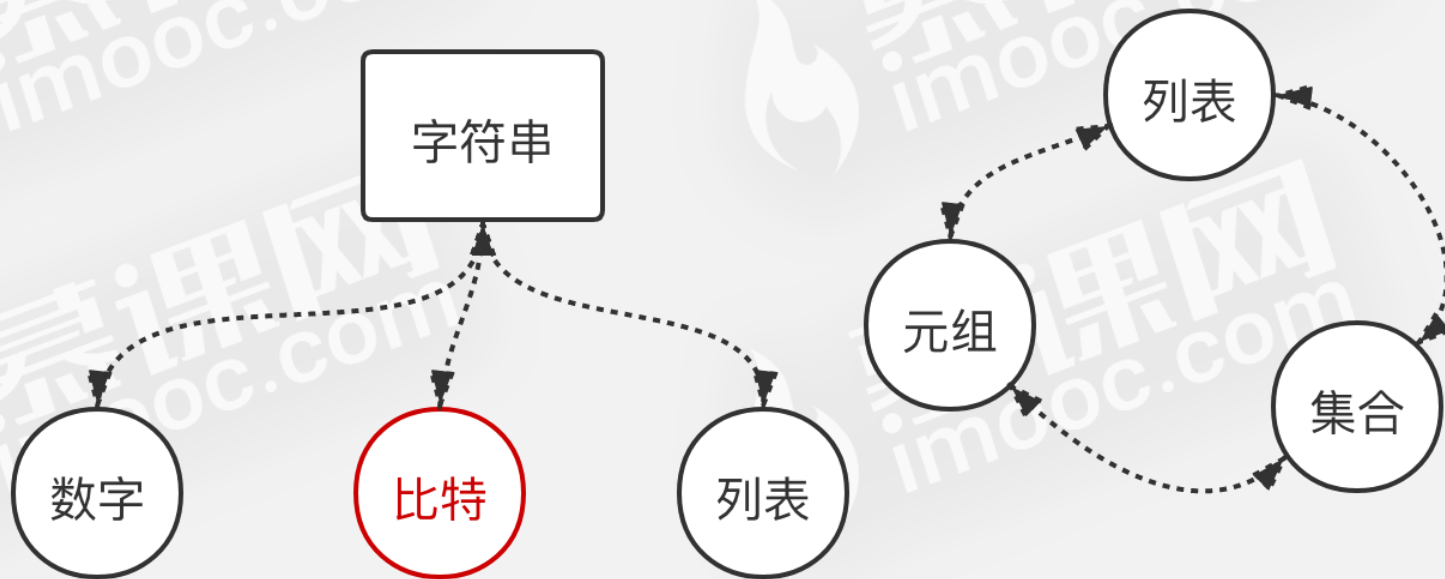


集合的操作



数据类型间的转换

为何要做
类型转换



条件语句

什么是条件语句

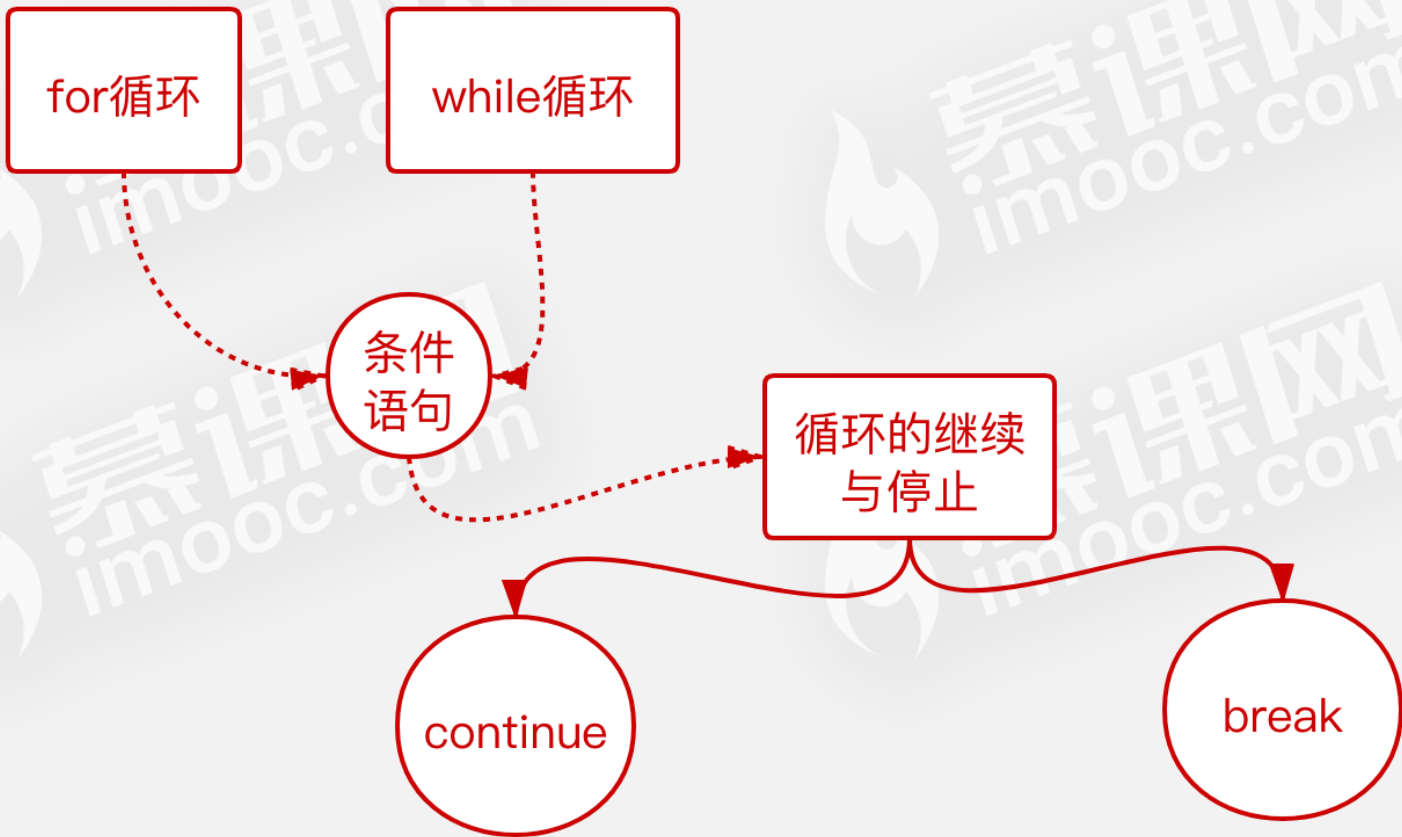
条件语句的关键词们

if

elif

else

循环语句



函数

函数的定义

函数结果的
返回

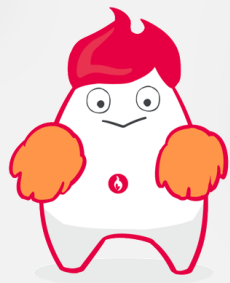
函数的参数

灵活性较大

全局与局部
变量

递归函数与
匿名函数

初识集合



本节课内容

- ◆ 什么是集合
- ◆ 集合与列表的区别
- ◆ 集合的创建方法



什么是集合

- ◆ 集合 (set) 是一个无序的不重复元素序列
- ◆ 常用来对两个列表进行交并差的处理性
- ◆ 集合与列表一样，支持所有数据类型

`{ 'name' , 1, 'xiaomu' }`

集合与列表的区别

功能	列表	集合
顺序	有序	无序
内容	可重复	不可重复
功能	用于数据的使用	用于数据的交集并集差集的获取
索引	有索引	无索引
符号	[] [1,2,3]	{ } {1, 2, 3}

集合的创建

- ◆ 通过 set函数来创建集合，不能使用{} 来创建空集合

`a_set = set()`

-> 空集合

`a_set = set([1, 2, 3])`

-> 传入列表或元组

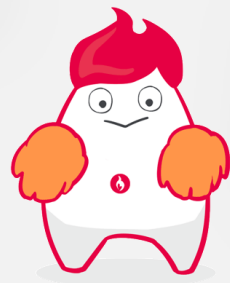
`b_set = {1, 2, 3}`

-> 传入元素

`c_set = {}`

-> X

集合的增删改



本节课内容

- ◆集合的add函数
- ◆集合的update函数
- ◆集合的remove函数
- ◆集合的clear函数
- ◆用del 删除集合
- ◆重要说明



add的功能

- ◆ 用于集合中添加一个元素，如果集合中已存在该元素则该函数不执行

add的用法

用法:

`set.add(item)`

参数:

item: 要添加到集合中的元素

返回值:

无返回值

```
In [10]: a_set = set()
```

```
In [11]: a_set.add('dewei')
```

```
In [12]: a_set
```

```
Out[12]: {'dewei'}
```


update的功能

- ◆ 加入一个新的集合（或列表，元组），如新集合内的元素在原集合中存在则无视

update的用法

用法:

`set.update(iterable)`

参数:

iterable: 一个集合或列表或元组

返回值:

无返回值, 直接作用于原集合

```
In [13]: a_set = set()
```

```
In [14]: a_set.update([3,4,5])
```

```
In [15]: a_set
```

```
Out[15]: {3, 4, 5}
```

remove的功能

- ◆ 将集合中的某个元素删除，如元素不存在将会报错

remove的用法

用法:

`set.remove(item)` # 注意是元素不是索引

参数:

`item`: 当前集合中的一个元素

返回值:

无返回值, 直接作用于原集合

```
In [19]: a_set = {1,2,3}
```

```
In [20]: a_set.remove(3)
```

```
In [21]: a_set
```

```
Out[21]: {1, 2}
```

clear的功能

◆ 清空当前集合中的所有元素

clear的用法

用法:

`set.clear()`

参数:

无

返回值:

无返回值, 直接作用于原集合

```
In [22]: a_set = {1,2,3}
```

```
In [23]: a_set.clear()
```

```
In [24]: a_set
```

```
Out[24]: set()
```

del 删除集合

举例：

```
a_set = {1, 2, 3}
```

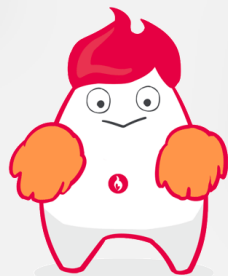
```
del a_set
```

```
print(a_set) # 报错
```

重要说明

- ◆ 集合无法通过索引获取元素
- ◆ 集合无获取元素的任何方法
- ◆ 集合只是用来处理列表或元组的一种临时类型，他不适合存储与传输

集合的差集—difference函数



本节课内容

- ◆ 什么是差集
- ◆ difference的功能
- ◆ difference的用法



什么是差集

◆ a, b两个集合，由所有属于a且不属于b的元素组成的集合，叫做a与b的差集



difference的功能

- ◆ 返回集合的差集，即返回的集合元素包含在第一个集合中，但不包含在第二个集合(方法的参数)中

difference的用法

用法:

a_set.difference(b_set)

参数:

b_set: 当前集合需要对比的集合

返回值:

返回原始集合于对比集合的差集
(即a_set与b_set的差集)

```
In [26]: a_set = {'name', 'xiaomu', 'xiaoming'}
```

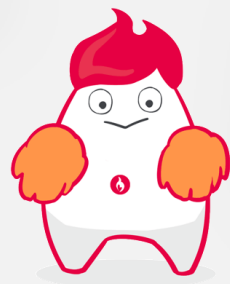
```
In [27]: b_set = {'xiaoming', 'xiaogang', 'xiaohong'}
```

```
In [28]: a_diff = a_set.difference(b_set)
```

```
In [29]: a_diff
```

```
Out[29]: {'name', 'xiaomu'}
```

集合的交集—intersection函数



本节课内容

- ◆ 什么是交集
- ◆ intersection的功能
- ◆ intersection的用法



什么是交集

- ◆ a, b两个集合分别拥有的相同的元素集, 称为a与b的交集



intersection的功能

- ◆ 返回两个或更多集合中都包含的元素，即交集

intersection的用法

用法:

a_set.intersection(b_set...)

参数:

b_set...: 与当前集合对比的1
或多个集合

返回值:

返回原始集合与对比集合的交集

```
In [32]: a_set = {'name', 'xiaomu', 'xiaoming'}
```

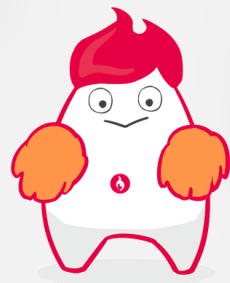
```
In [33]: b_set = {'xiaoming', 'xiaogang', 'xiaohong'}
```

```
In [34]: a_inter = a_set.intersection(b_set)
```

```
In [35]: a_inter
```

```
Out[35]: {'xiaoming'}
```

集合的交集—union函数



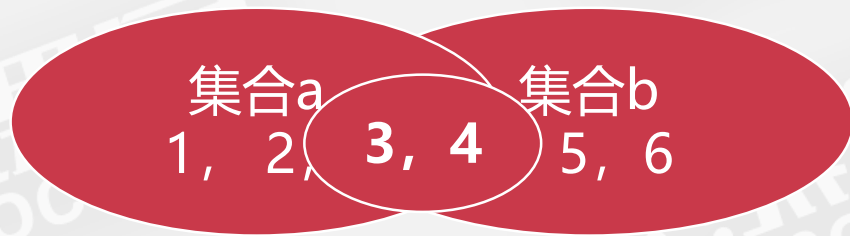
本节课内容

- ◆ 什么是并集
- ◆ union的功能
- ◆ union的用法



什么是并集

- ◆ a, b两个集合中所有的元素（去掉重复）即为a与b的并集



union的功能

- ◆ 返回多个集合的并集，即包含了所有集合的元素，重复的元素只会出现一次

union的用法

用法:

`a_set.union(b_set...)`

参数:

`b_set...`: 与当前集合对比
的1或多个集合

返回值:

返回原始集合与对比集合的并集

union的用法

```
In [36]: a_set = {'name', 'xiaomu', 'xiaoming'}
```

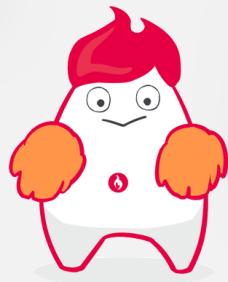
```
In [37]: b_set = {'xiaoming', 'xiaogang', 'xiaohong'}
```

```
In [38]: un = a_set.union(b_set)
```

```
In [39]: un
```

```
Out[39]: {'name', 'xiaogang', 'xiaohong', 'xiaoming', 'xiaomu'}
```


集合的isdisjoint函数



本节课内容

- ◆ isdisjoint的功能
- ◆ isdisjoint的用法



isdisjoint的功能

- ◆ 判断两个集合是否包含相同的元素，如果没有返回 True，否则返回 False

isdisjoint的用法

用法:

```
a_set.isdisjoint(b_set)
```

参数:

b_set: 与当前集合用来判断的集合

返回值:

返回一个布尔值 True 或 False

isdisjoint的用法

```
In [57]: a_set = {'name', 'xiaomu', 'xiaoming'}
```

```
In [58]: b_set = {'xiaoming', 'xiaogang', 'xiaohong'}
```

```
In [59]: result = a_set.isdisjoint(b_set)
```

```
In [60]: result
```

```
Out[60]: False
```