



python

```
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
    turtle.circle(40, 80/2)
    turtle.fd(40)
    turtle.circle(16, 180)
    turtle.fd(40 * 2/3)
```

Python语言程序设计

集合类型及操作



嵩天
北京理工大学



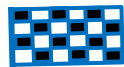


单元开篇

集合类型及操作



- 集合类型定义
- 集合操作符
- 集合处理方法
- 集合类型应用场景





集合类型定义

集合类型的定义

集合是多个元素的无序组合

- 集合类型与数学中的集合概念一致
- 集合元素之间无序，每个元素唯一，不存在相同元素
- 集合元素不可更改，不能是可变数据类型 **为什么？**

集合类型的定义

集合是多个元素的无序组合

- 集合用大括号 {} 表示，元素间用逗号分隔
- 建立集合类型用 {} 或 set()
- 建立空集合类型，必须使用set()

集合类型的定义

```
>>> A = {"python", 123, ("python",123)} #使用{}建立集合
{123, 'python', ('python', 123)}

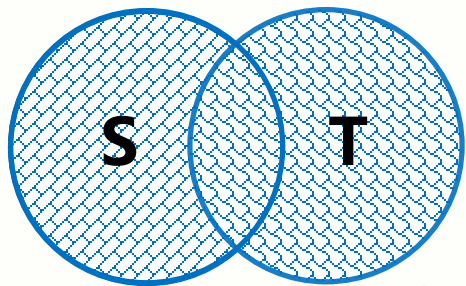
>>> B = set("pypy123") #使用set()建立集合
{'1', 'p', '2', '3', 'y'}

>>> C = {"python", 123, "python",123}
{'python', 123}
```

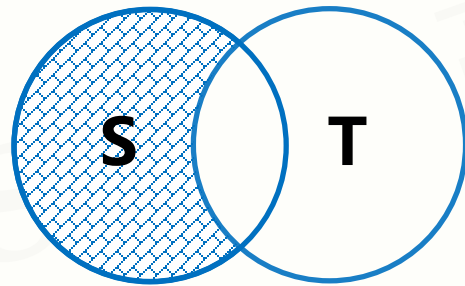



集合操作符

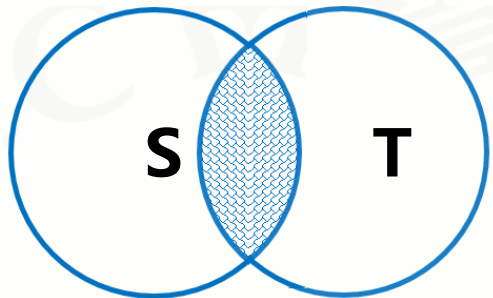
集合间操作



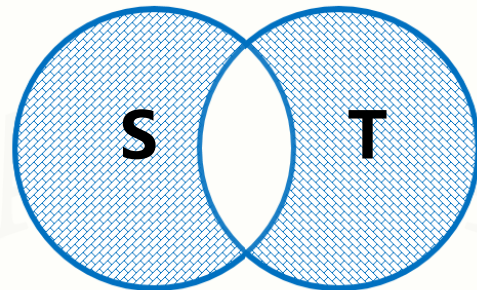
$S \cup T$
并



$S - T$
差



$S \cap T$
交



$S \Delta T$
补

集合操作符

6个操作符

操作符及应用	描述
$S \mid T$	返回一个新集合，包括在集合S和T中的所有元素
$S - T$	返回一个新集合，包括在集合S但不在T中的元素
$S \& T$	返回一个新集合，包括同时在集合S和T中的元素
$S \wedge T$	返回一个新集合，包括集合S和T中的非相同元素
$S \leq T$ 或 $S < T$	返回True/False，判断S和T的子集关系
$S \geq T$ 或 $S > T$	返回True/False，判断S和T的包含关系

集合操作符

4个增强操作符

操作符及应用	描述
$S = T$	更新集合S, 包括在集合S和T中的所有元素
$S -= T$	更新集合S, 包括在集合S但不在T中的元素
$S \&= T$	更新集合S, 包括同时在集合S和T中的元素
$S \wedge= T$	更新集合S, 包括集合S和T中的非相同元素

集合类型的定义

```
>>> A = {"p", "y", 123}
```

```
>>> B = set("pypy123")
```

```
>>> A-B
```

```
{123}
```

```
>>> B-A
```

```
{'3', '1', '2'}
```

```
>>> A&B
```

```
{'p', 'y'}
```

```
>>> A|B
```

```
{'1', 'p', '2', 'y', '3', 123}
```

```
>>> A^B
```

```
{'2', 123, '3', '1'}
```



集合处理方法

集合处理方法

操作函数或方法	描述
<code>S.add(x)</code>	如果x不在集合S中，将x增加到S
<code>S.discard(x)</code>	移除S中元素x，如果x不在集合S中，不报错
<code>S.remove(x)</code>	移除S中元素x，如果x不在集合S中，产生KeyError异常
<code>S.clear()</code>	移除S中所有元素
<code>S.pop()</code>	随机返回S的一个元素，更新S，若S为空产生KeyError异常

集合处理方法

操作函数或方法	描述
<code>S.copy()</code>	返回集合S的一个副本
<code>len(S)</code>	返回集合S的元素个数
<code>x in S</code>	判断S中元素x, x在集合S中, 返回True, 否则返回False
<code>x not in S</code>	判断S中元素x, x不在集合S中, 返回False, 否则返回True
<code>set(x)</code>	将其他类型变量x转变为集合类型

集合处理方法

```
>>> A = {"p", "y", 123}
>>> for item in A:
    print(item, end="")
p123y

>>> A
{'p', 123, 'y'}
```

```
>>> try:
    while True:
        print(A.pop(), end="")
    except:
        pass

p123y

>>> A
set()
```



集合类型应用场景



集合类型应用场景

包含关系比较

```
>>> "p" in {"p", "y", 123}
```

```
True
```


```
>>> {"p", "y"} >= {"p", "y", 123}
```

```
False
```

集合类型应用场景

数据去重：集合类型所有元素无重复

```
>>> ls = ["p", "p", "y", "y", 123]
>>> s = set(ls)      # 利用了集合无重复元素的特点
{'p', 'y', 123}
>>> lt = list(s)     # 还可以将集合转换为列表
['p', 'y', 123]
```



单元小结

集合类型及操作

- 集合使用{}和set()函数创建
- 集合间操作：交(&)、并(|)、差(-)、补(^)、比较(>=<)
- 集合类型方法：.add()、.discard()、.pop()等
- 集合类型主要应用于：包含关系比较、数据去重



