


Python基础语法精讲

嵩天

组合数据类型使用方法

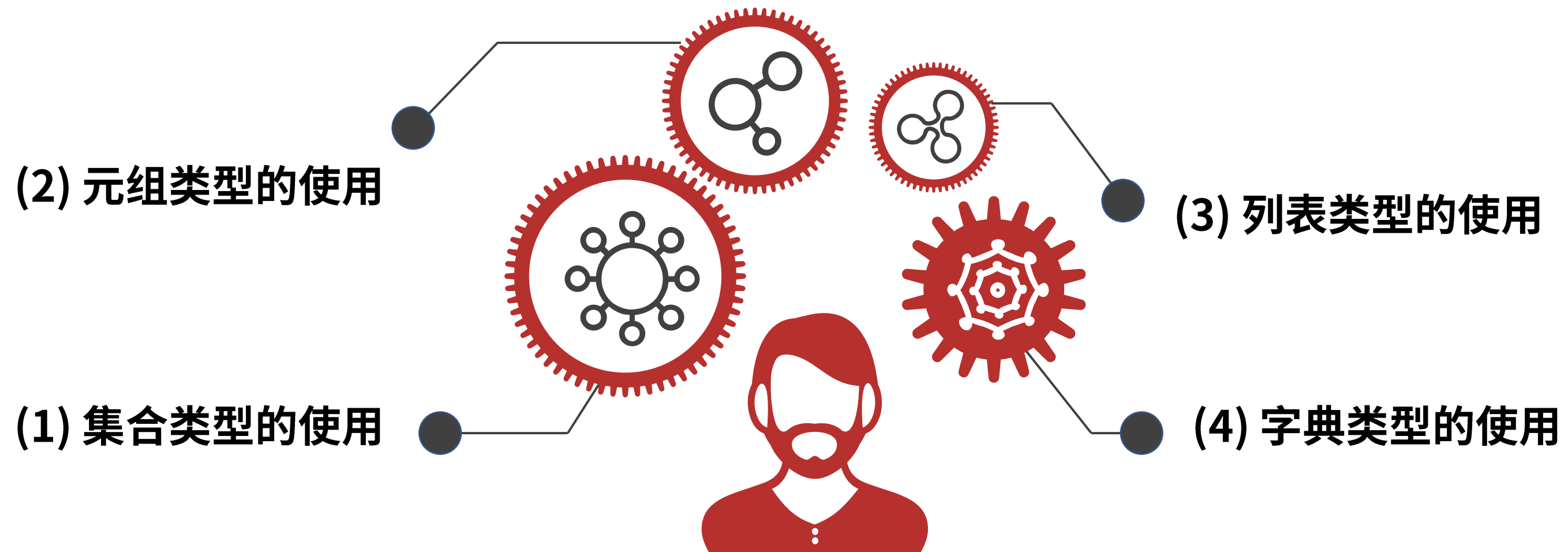
高天



组合数据类型使用方法

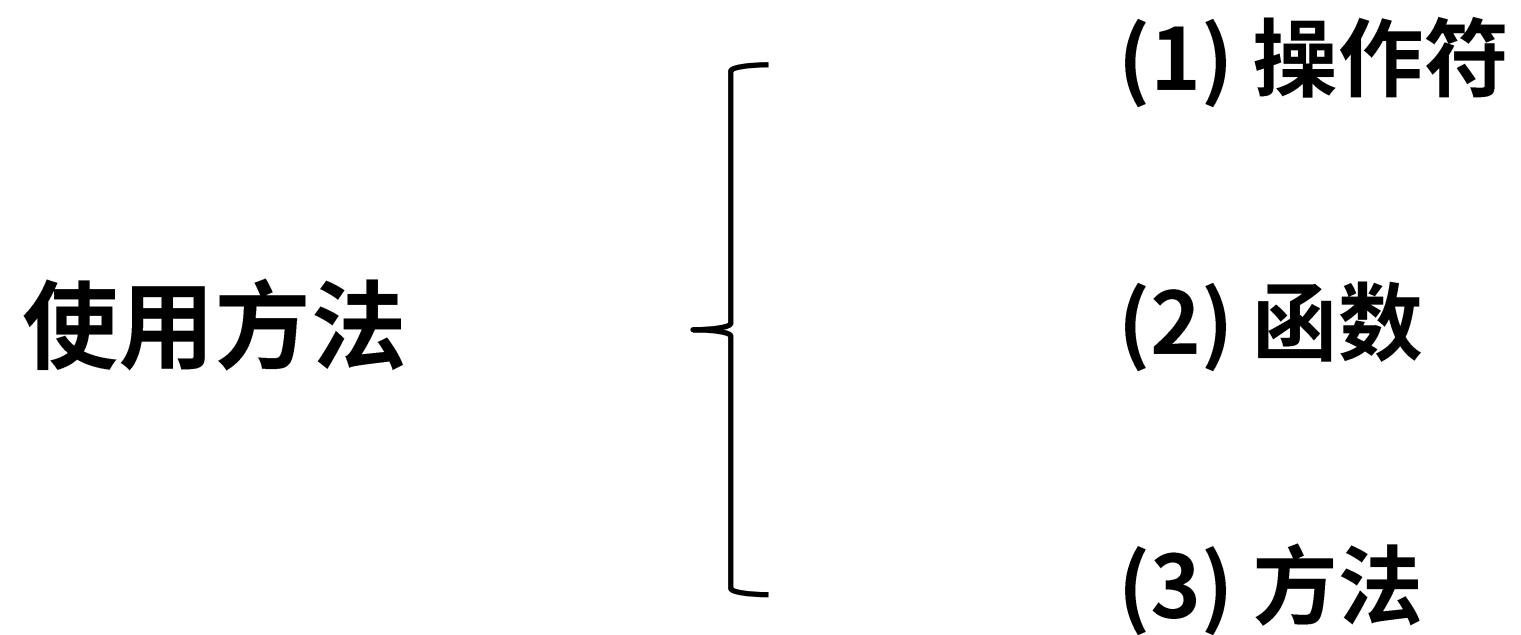
单元开篇

单元开篇



组合数据类型使用方法

单元开篇



组合数据类型使用方法

组合数据类型使用方法

集合类型 的使用

集合类型的使用

- 操作符：包含(in)、交(&)、并(|)、差(-)、补(^)、比较(>=<)
- 函数：len()、set()
- 方法：
 - .add()、.remove()、.discard()、.clear()、.pop()、.copy()
 - .intersection()、.union()、.difference()、symmetric_difference()
 - .intersection_update()、.update()、difference_update()、.symmetric_difference_update()
 - .isdisjoint() .issubset()、issuperset()

集合类型的操作符

基本操作符（交并差补）

操作符	含义	示例
in	元素判断	x in S
not in	元素判断	x not in S
&	集合的交集，返回一个新集合	S & T
	集合的并集，返回一个新集合	S T
-	集合的差集，返回一个新集合	S - T
^	集合的补集，返回一个新集合	S ^ T

集合类型的操作符

比较操作符

操作符	含义	示例
<	真子集判断	$S < T$
<=	子集判断	$S \leq T$
>	真超集判断	$S > T$
>=	超集判断	$S \geq T$
==	全相同判断	$S == T$
!=	不相同判断	$S != T$

集合类型的函数

内置操作函数

函数	含义	示例
len(x)	返回集合的元素个数	len(S)
set(x)	转换组合类型，创建一个集合	set([1,1,1,2,2,2]) {1, 2}

集合类型的方法

集合元素维护类方法

方法及使用	含义	示例
<code>.add(x)</code>	增加x到集合	<code>S.add(x)</code>
<code>.remove(x)</code>	删除S中元素x，如果x不存在，产生KeyError	<code>S.remove(x)</code>
<code>.discard(x)</code>	删除S中元素x，如果x不存在，不报错	<code>S.discard(x)</code>
<code>.clear()</code>	删除S中所有元素	<code>S.clear()</code>
<code>.pop()</code>	随机返回S中一个元素，如果S为空，产生KeyError	<code>S.pop()</code>
<code>.copy()</code>	复制集合S，产生一个副本	<code>S.copy()</code>

集合间运算类方法(1)

方法及使用	含义	示例	
<code>.intersection(x)</code>	集合的交集，返回一个新集合	<code>S.intersection(T)</code>	不更新S
<code>.union(x)</code>	集合的并集，返回一个新集合	<code>S.union(T)</code>	不更新S
<code>.difference(x)</code>	集合的差集，返回一个新集合	<code>S.difference(T)</code>	不更新S
<code>.symmetric_difference()</code>	集合的补集，返回一个新集合	<code>S.symmetric_difference(T)</code>	

集合间运算类方法(2)

方法及使用	含义	示例
<code>.intersection_update(x)</code>	集合的交集，更新原集合	<code>S.intersection_update(T)</code> 更新S
<code>.update(x)</code>	集合的并集，更新原集合	<code>S.update(T)</code>
<code>.difference_update (x)</code>	集合的差集，更新原集合	<code>S.difference_update(T)</code>
<code>.symmetric_difference_update ()</code>	集合的补集，更新原集合	<code>S.symmetric_difference_update(T)</code>

集合类型的方法

集合间比较类方法

方法及使用	含义	示例
<code>.isdisjoint(x)</code>	无关判断，两个集合之间无共同元素则返回True	<code>S.isdisjoint(T)</code>
<code>.issubset(x)</code>	子集判断，如果x是集合的子集则返回True	<code>S.issubset(T)</code>
<code>.issuperset(x)</code>	超集判断，如果x是集合的超集则返回True	<code>S.issuperset(T)</code>

集合类型的应用场景

包含关系比较

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

```
True
```

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

```
False
```

数据去重

```
>>> 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]
```

集合类型的使用

- 操作符：包含(in)、交(&)、并(|)、差(-)、补(^)、比较(>=<)
- 函数：len()、set()
- 方法：
.add()、.remove()、.discard()、.clear()、.pop()、.copy()
.intersection()、.union()、.difference()、.symmetric_difference()
.intersection_update()、.update()、.difference_update()、.symmetric_difference_update()
.isdisjoint()、.issubset()、.issuperset()

组合数据类型使用方法

元组类型 的使用

元组类型的使用

元组类型的使用纵览

- 操作符: in、+、*、比较(>=<)
- 函数: len()、tuple()、min()、max()
- 方法: .index()、.count()

元组类型的操作符

基本操作符

操作符	含义	示例
in	元素判断	1 in (1,2,3)
not in	元素判断	1 not in (1,2,3)
+	连接多个元组，返回一个新元组	(1,2,3) + (4,5,6) + (7,8)
*	重复元组多次，返回一个新元组	(1,2,3)*3

元组类型的操作符

比较操作符

操作符	含义	示例	
<	按照顺序，逐个元素比较 只要元素比较得出True/False，则返回结果 比较时，元素间要有可比性	(11, 2) < (13, 1)	结果 True
<=		(11, 2) <= (11, 1, 2)	结果 False
>		(11, 2) > (10, "abc")	结果 True
>=		(11, 2) >= (11, 2)	结果 True
==		(11, 2) == (11, 2, 1)	结果 False
!=		(11, 2) != (1, 2)	结果 True

元组类型的函数

内置操作函数

函数	含义	示例
len(x)	返回元组的元素个数	len(S)
tuple(x)	转换组合类型，创建一个元组	tuple([1,1,1,2,2,2]) 结果 (1, 1, 1, 2, 2, 2)
min(x)	返回元组中最小的元素	min((1,1,1,2,2,2))
max(x)	返回元组中最大的元素	max((1,1,1,2,2,2))

元组类型的方法

方法及使用	含义	示例
<code>.index(x)</code>	返回元组中第一次出现x的位置	<code>tp.index(x)</code>
<code>.count(x)</code>	返回元组中出现x的总次数	<code>tp.count(x)</code>

元组类型无增删改查需求，只能索引、切片、查询和统计

元组类型的使用

元组类型的使用纵览

- 操作符: in、+、*、比较(>=<)
- 函数: len()、tuple()、min()、max()
- 方法: .index()、.count()

组合数据类型使用方法

列表类型 的使用

列表类型的使用

列表类型的使用纵览

- 操作符：in、del、+、*、比较(>=<)
- 函数：len()、list()、min()、max()
- 方法：.append()、.insert()、.extend()、.remove()、.clear()
.copy()、.pop()、.reverse()、.index()、.count()、.sort()

列表类型的操作符

基本操作符

操作符	含义	示例
in	元素判断	1 in [1,2,3]
not in	元素判断	1 not in [1,2,3]
del	删除列表元素或列表片段	del ls[2] 或 del[: -2]
+	连接多个列表，返回一个新列表	[1,2,3] + [4,5,6] + [7,8]
*	重复列表多次，返回一个新列表	[1,2,3]*3

列表类型的操作符

比较操作符

操作符	含义	示例
<	按照顺序，逐个元素比较 只要元素比较得出True/False，则返回结果 比较时，元素间要有可比性	[11, 2] < [13, 1] 结果 True
<=		[11, 2] <= [11, 1, 2] 结果 False
>		[11, 2] > [10, "abc"] 结果 True
>=		[11, 2] >= [11, 2] 结果 True
==		[11, 2] == [11, 2, 1] 结果 False
!=		[11, 2] != [1, 2] 结果 True

列表类型的函数

内置操作函数

函数	含义	示例
len(x)	返回列表的元素个数	len(ls)
list(x)	转换组合类型，创建一个列表	list((1,1,1,2,2,2)) 结果 [1, 1, 1, 2, 2, 2]
min(x)	返回列表中最小的元素	min([1,1,1,2,2,2])
max(x)	返回列表中最大的元素	max([1,1,1,2,2,2])

列表类型的方法

列表的增删方法

方法及使用	含义	示例
<code>.append(x)</code>	在列表最后增加一个元素x	<code>ls.append(x)</code>
<code>.insert(x)</code>	在列表第i位置增加元素x	<code>ls.insert(x)</code>
<code>.extend(lt)</code>	在列表最后增加一个新列表	<code>ls.extend(lt)</code>
<code>.remove()</code>	删除列表中第一次出现的元素x	<code>ls.remove(x)</code>
<code>.clear()</code>	删除列表中所有元素	<code>ls.clear()</code>

列表类型的方法

列表的维护查询

方法及使用	含义	示例
<code>.copy()</code>	拷贝列表中所有元素，生成一个新列表	<code>lt = ls.copy()</code>
<code>.pop(i)</code>	将列表第i位置元素取出并删除该元素	<code>ls.pop(i)</code>
<code>.reverse()</code>	列表中顺序元素反转	<code>ls.reverse()</code>
<code>.index(x)</code>	返回列表中第一次出现x的位置	<code>ls.index(x)</code>
<code>.count(x)</code>	返回列表中出现x的总次数	<code>ls.count(x)</code>
<code>.sort()</code>	对列表进行排序，默认是值递增	<code>ls.sort()</code>

列表类型的使用

列表类型的使用纵览

- 操作符：in、del、+、*、比较(>=<)
- 函数：len()、list()、min()、max()
- 方法：.append()、.insert()、.extend()、.remove()、.clear()
.copy()、.pop()、.reverse()、.index()、.count()、.sort()

列表类型的使用

列表使用小练习 (1)

- 定义空列表lt
 - 向lt新增5个元素
 - 修改lt中第2个元素
 - 向lt中第2个位置增加一个元素
 - 从lt中第1个位置删除一个元素
 - 删除lt中第1-3位置元素
- ```
>>> lt = []
>>> lt += [1,2,3,4,5]
>>> lt[2] = 6
>>> lt.insert(2, 7)
>>> del lt[1]
>>> del lt[1:4]
```



# 列表类型的使用

## 列表使用小练习 (2)

- 判断lt中是否包含数字0
- 向lt新增数字0
- 返回数字0所在lt中的索引
- lt的长度
- lt中最大元素
- 清空lt

```
>>> 0 in lt
```

```
>>> lt.append(0)
```

```
>>> lt.index(0)
```

```
>>> len(lt)
```

```
>>> max(lt)
```

```
>>> lt.clear()
```

# 列表类型的应用场景

## 列表适用于组织数据

- 列表用来组织数据，非常灵活，它是最常用的组合数据类型
- 列表可用于表示一组有序数据或一组顺序无关数据，进而操作它们
- 列表将用于表达一二维数据
- 列表计算性能并不高，对于大规模数据，建议采用第三方数据类型，如ndarray

组合数据类型使用方法

# 字典类型 的使用

# 字典类型的使用

## 字典类型的使用纵览

- 操作符: `in`、`del`、`==`、`!=`
- 函数: `len()`、`dict()`、`iter()`
- 方法: `.items()`、`.keys()`、`.values()`、`.pop()`、`.popitem()`  
`.update()`、`.clear()`、`.copy()`、`.get()`

# 字典类型的操作符

## 基本操作符

| 操作符           | 含义           | 示例                                  |
|---------------|--------------|-------------------------------------|
| <b>in</b>     | 根据键的元素包含判断   | 'a' in {'a': 1, 'b': 2, 'c': 3}     |
| <b>not in</b> | 根据键的元素包含判断   | 'a' not in {'a': 1, 'b': 2, 'c': 3} |
| <b>del</b>    | 根据键删除字典中单个元素 | del d['b']                          |
| <b>==</b>     | 判断两个字典相同     | d == {'a': 1, 'b': 2, 'c': 3}       |
| <b>!=</b>     | 判断两个字典相同     | d != {'a': 1, 'b': 2, 'c': 3}       |

# 字典类型的函数

## 内置操作函数

| 函数      | 含义               | 示例      |
|---------|------------------|---------|
| len(d)  | 返回字典的元素个数        | len(d)  |
| dict()  | 创建一个字典，一般用于创建空字典 | dict()  |
| iter(d) | 根据字典d的键形成一个迭代类型  | iter(d) |

# 字典类型的方法

## 字典的元素获取（键值对）

| 方法及使用                   | 含义                               | 示例                       |
|-------------------------|----------------------------------|--------------------------|
| <code>.items()</code>   | 返回字典所有键值对，键值对采用(key,value)元组形式返回 | <code>d.items()</code>   |
| <code>.keys()</code>    | 返回字典的键                           | <code>d.keys()</code>    |
| <code>.values()</code>  | 返回字典的值                           | <code>d.values()</code>  |
| <code>.pop(k)</code>    | 取出特定键k对应的值，并删除键值对                | <code>d.pop('a')</code>  |
| <code>.popitem()</code> | 随机从字典中取出一个键值对，以(key,value)元组形式返回 | <code>d.popitem()</code> |

# 字典类型的方法

## 字典的维护方法

| 方法及使用                         | 含义                     | 示例                         |
|-------------------------------|------------------------|----------------------------|
| <code>.update(t)</code>       | 扩展其他字典t的内容到当前字典，键重复的替换 | <code>d.update(t)</code>   |
| <code>.clear()</code>         | 删除字典所有元素               | <code>d.clear()</code>     |
| <code>.copy()</code>          | 拷贝字典中所有元素，生成一个新字典      | <code>t = d.copy()</code>  |
| <code>.get(k, default)</code> | 键k存在则返回对应值，否则返回default | <code>d.get('a', 4)</code> |



# 字典类型的使用

## 字典类型的使用纵览

- 操作符: `in`、`del`、`==`、`!=`
- 函数: `len()`、`dict()`、`iter()`
- 方法: `.items()`、`.keys()`、`.values()`、`.pop()`、`.popitem()`  
`.update()`、`.clear()`、`.copy()`、`.get()`

组合数据类型使用方法

# 单元小结

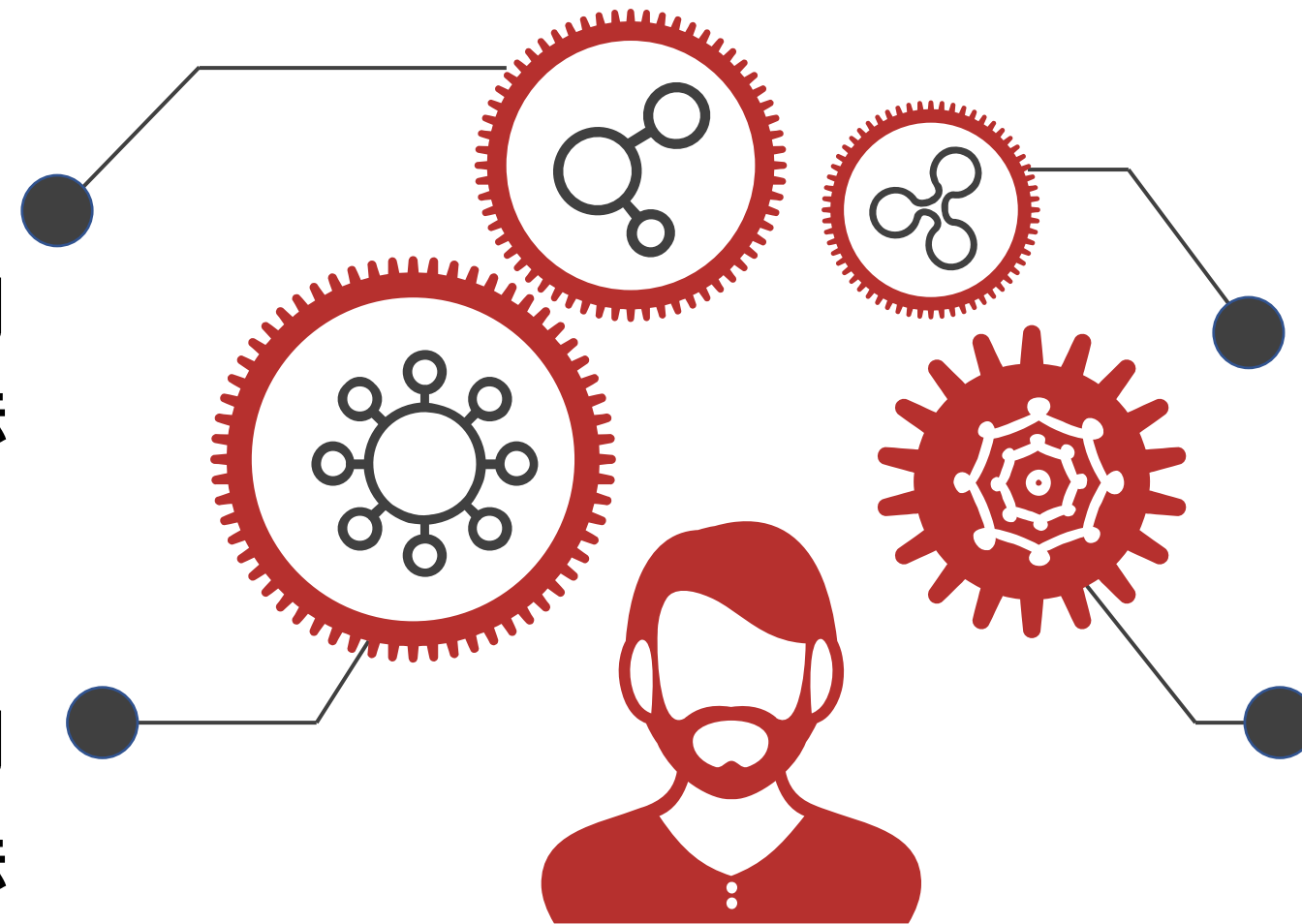
# 单元小结

(2) 元组类型的使用  
操作符、函数和方法

(1) 集合类型的使用  
操作符、函数和方法

(3) 列表类型的使用  
操作符、函数和方法

(4) 字典类型的使用  
操作符、函数和方法



## 组合数据类型使用方法

# 集合类型的使用

- 操作符：包含(in)、交(&)、并(|)、差(-)、补(^)、比较(>=<)
- 函数：len()、set()
- 方法：
  - .add()、.remove()、.discard()、.clear()、.pop()、.copy()
  - .intersection()、.union()、.difference()、symmetric\_difference()
  - .intersection\_update()、.update()、difference\_update()、.symmetric\_difference\_update()
  - .isdisjoint() .issubset()、issuperset()

# 元组类型的使用

## 元组类型的使用纵览

- 操作符: in、+、\*、比较(>=<)
- 函数: len()、tuple()、min()、max()
- 方法: .index()、.count()

# 列表类型的使用

## 列表类型的使用纵览

- 操作符：in、del、+、\*、比较(>=<)
- 函数：len()、list()、min()、max()
- 方法：.append()、.insert()、.extend()、.remove()、.clear()  
.copy()、.pop()、.reverse()、.index()、.count()、.sort()

# 字典类型的使用

## 字典类型的使用纵览

- 操作符: `in`、`del`、`==`、`!=`
- 函数: `len()`、`dict()`、`iter()`
- 方法: `.items()`、`.keys()`、`.values()`、`.pop()`、`.popitem()`  
`.update()`、`.clear()`、`.copy()`、`.get()`

 Python ▶ 123

# Thank you