文章目录

1 六大数据类型

- Number（数字）
- String（字符串）
- Tuple（元组）
- List（列表）
- Dictionary（字典）
- Set（集合）
- **不可变数据**（3 个）：Number（数字）、String（字符串）、Tuple（元组）
- **可变数据**（3 个）：List（列表）、Dictionary（字典）、Set（集合）

2 Number（数字）

三种不同的数值类型：

- **int**(整型)：通常被称为是整型或整数，是正或负整数，不带小数点。布尔(bool)是整型的子类型。
- **float**(浮点型)：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 100 = 250）

- **complex**(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

数字类型转换：

- `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

常用数学函数：

函数	描述
abs(x)	返回数字的绝对值，如abs(-10) 返回 10
fabs(x)	返回数字的绝对值，如math.fabs(-10) 返回10.0
ceil(x)	返回数字的上入整数，如math.ceil(4.1) 返回 5
floor(x)	返回数字的下舍整数，如math.floor(4.9)返回 4
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。
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。

3 String（字符串）

Python中的字符串用单引号 `'` 或双引号 `"` 括起来，同时使用反斜杠 `\` 转义特殊字符。

常用字符串运算符：

操作符	描述	实例
+	字符串连接	a + b 输出结果： ab
*	重复输出字符串	a*2 输出结果：aa
[]	通过索引获取字符串中字符	a=‘Hello’，a[1] 输出结果：e
[:]	截取字符串中的一部分，遵循左闭右开原则，str[0:2] 是不包含第 3 个字符的	a=‘Hello’，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 Hello 输出结果 True
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 Hello 输出结果 True

字符串格式化：

```
name = '小袁'
age = 20
# 语法一：%
print('我的名字是：%s ,年龄是：%d' % (name,age)) # 我的名字是：小袁 ,年龄是：20
#语法二：f'{表达式}'
print(f'我的名字是：{name},我的年龄是：{age}') # 我的名字是：小袁,我的年龄是：20
```

4 Tuple（元组）

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，并且元组中的元素不能改变！

```
tup1 = ('hello', 'world', 1, 2)
print(tup1) # ('hello', 'world', 1, 2)
print(type(tup1)) # <class 'tuple'>
```

遍历元组：

```
tup1 = ('hello', 'world', 1, 2)

for i in tup1:
    print(i,end=" ")
# hello world 1 2
```

常用运算符：

操作符	描述	实例
len()	计算元素个数	len(tup1)，输出结果：4
+	连接	tup1 + (3,4)，输出结果：（‘hello’，‘world’, 1, 2, 3, 4)
*	复制	（‘Hi!’,) * 4，输出结果：（‘Hi!’，‘Hi!’，‘Hi!’，‘Hi!’）
in	元素是否存在	3 in (1, 2, 3)，输出结果：True
[]	读取第几个元素	[0]，输出结果：hello
[:]	截取字符串中的一部分，遵循左闭右开原则	[0:2]，输出结果：（‘hello’，‘world’）

5 List（列表）

列表是写在方括号 `[]` 之间、用逗号分隔开的元素列表。列表中元素的类型可以不相同，它支持数字，字符串甚至可以包含列表（所谓嵌套）。列表中的元素是可以改变的！

修改列表：

```
a = [1, 2, 3, 4, 5]
# 下表索引的方式修改
a[0] = 9
print(a) # [9, 2, 3, 4, 5]

# append()方法：追加列表
a.append(6)
print(a) # [9, 2, 3, 4, 5, 6]

# del 语句来删除列表的的元素
del a[0]
print(a) # [2, 3, 4, 5, 6]
```

嵌套列表：

```
a = [1, 2, 3, 4, 5]
b = ['a', 'b', 'c']

x = [a, b]
print(x) # [[1, 2, 3, 4, 5], ['a', 'b', 'c']]
print(x[0]) # [1, 2, 3, 4, 5]
print(x[0][1]) # 2
```

遍历列表：

```
a = [1, 2, 3, 4, 5]

for i in a:
    print(i,end=" ")
# 1 2 3 4 5
```

常用运算符同元组!

常用方法:

方法名	描述
list.append(obj)	在列表末尾添加新的对象
list.count(obj)	统计某个元素在列表中出现的次数
list.index(obj)	从列表中找出某个值第一个匹配项的索引位置
list.insert(index, obj)	将对象从对应索引位置插入列表
list.pop([index=-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
list.reverse()	反转列表中元素
list.sort(key=None, reverse=False)	对原列表进行排序
list.clear()	清空列表
list.copy()	复制列表

6 Dictionary（字典）

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, key3 : value3 }
```

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

访问字典的值:

```
dict = {'Name': '小明', 'Age': 20}
print(dict) # {'Name': '小明', 'Age': 20}

print (dict['Name']) # 小明
print (dict['Age']) # 20
```

修改字典:

```
dict = {'Name': '小明', 'Age': 20}

dict['Name'] = '小黑'
dict['Age'] = 22

print(dict) # {'Name': '小黑', 'Age': 22}
```

遍历字典:

```
dict = {'Name': '小明', 'Age': 20}

#遍历键
for key in dict.keys():
    print(key)
"""
Name
Age
"""

# 遍历值
for value in dict.values():
    print(value)
"""
小明
20
"""
```

7 Set（集合）

集合可以使用大括号 `{}` 或者 `set()` 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。集合是一个无序的不重复元素序列，集合内的元素可以改变！

两种创建格式:

```
set1 = {'小黑',20,20}
print(set1) # {'小黑', 20} ；元素不重复只显示一个20

set2 = set('abcd')
print(set2) # {'c', 'b', 'd', 'a'} ; 元素没有顺序
```

修改集合:

```
set1 = {'小黑',20,20}

#add()：添加方法
set1.add('大学生')
print(set1) # {'大学生', '小黑', 20}

# update()：也可以添加元素，且参数可以是列表，元组，字典等
set1.update([1,2],[3,4])
print(set1) # {1, '大学生', 2, 3, 4, 20, '小黑'}

# remove()：移除元素
set1.remove('大学生')
print(set1) # {1, 2, 3, 4, 20, '小黑'}
```

遍历集合：

```
set1 = {'小黑',20,20}

for i in set1:
    print(i,end=" ")
# 20 小黑
```

8 数据类型转换函数

int()

> 将float、bool、str类型的数据转换为int类型。float类型转换为int类型时去除小数点后面的数；bool类型转换为int类型时False变为0、True变为1；str类型直接转换为int类型

案例：

```
# 定义float变量
f = 9.99
# 定义bool类型变量
b1 = False
b2 = True
# 定义str类型变量
s = '111'

# 使用int()函数
int1 = int(f)
int2 = int(b1)
int3 = int(b2)
int4 = int(s)

print("int1:",int1)
print("int1的类型是：",type(int1))
```

```
print("-"*10)

print("int2:",int2)
print("int2的类型是: ",type(int2))
print("int3:",int3)
print("int3的类型是: ",type(int3))
print("-"*10)

print("int3:",int4)
print("int3的类型是: ",type(int4))

'''
int1: 9
int1的类型是:  <class 'int'>
----------
int2: 0
int2的类型是:  <class 'int'>
int3: 1
int3的类型是:  <class 'int'>
----------
int3: 111
int3的类型是:  <class 'int'>
'''
```

bool()

将int、float、str类型的数据转换为bool类型。int类型转换为bool类型时0变为False、其他数据变为True；float类型转换为bool时0.0变为False、其他数据变为True；str类型转换为bool类型时不存在数据变为False、存在数据变为True。

案例:

```
# 定义int变量
i1 = 0
i2 = -1
i3 = 1
# 定义float变量
f1 = 0.0
f2 = -1.0
f3 = 1.0
# 定义str变量
s1 = ''
s2 = '0'
s3 = '-1'
s4 = '1'
s5 = 'A'

# 使用bool()函数
```

```
b1 = bool(i1)
b2 = bool(i2)
b3 = bool(i3)
b4 = bool(f1)
b5 = bool(f2)
b6 = bool(f3)
b7 = bool(s1)
b8 = bool(s2)
b9 = bool(s3)
b10 = bool(s4)
b11 = bool(s5)

print("b1:",b1)
print("b1的类型是: ",type(b1))
print("b2:",b2)
print("b2的类型是: ",type(b2))
print("b3:",b3)
print("b3的类型是: ",type(b3))
print("-"*10)

print("b4:",b4)
print("b4的类型是: ",type(b4))
print("b5:",b5)
print("b5的类型是: ",type(b5))
print("b6:",b6)
print("b6的类型是: ",type(b6))
print("-"*10)

print("b7:",b7)
print("b7的类型是: ",type(b7))
print("b8:",b8)
print("b8的类型是: ",type(b8))
print("b9:",b9)
print("b9的类型是: ",type(b9))
print("b10:",b10)
print("b10的类型是: ",type(b10))
print("b11:",b11)
print("b11的类型是: ",type(b11))

'''
b1: False
b1的类型是:  <class 'bool'>
b2: True
b2的类型是:  <class 'bool'>
b3: True
b3的类型是:  <class 'bool'>
----------
b4: False
b4的类型是:  <class 'bool'>
```

```
b5: True
b5的类型是: <class 'bool'>
b6: True
b6的类型是: <class 'bool'>
----------
b7: False
b7的类型是: <class 'bool'>
b8: True
b8的类型是: <class 'bool'>
b9: True
b9的类型是: <class 'bool'>
b10: True
b10的类型是: <class 'bool'>
b11: True
b11的类型是: <class 'bool'>
'''
```

float()

> 将int、bool、str类型的数据转换为float类型数据。int类型转换为float时在末尾添加小数位；bool类型转换为float时False变为0.0,、True变为1.0；str类型直接转换为float类型。

案例:

```
# 定义int变量
i1 = 1
i2 = -1
# 定义bool变量
b1 = False
b2 = True
# 定义str变量
s1 = '99'

# 使用float()函数
f1 = float(i1)
f2 = float(i2)
f3 = float(b1)
f4 = float(b2)
f5 = float(s1)

print("f1:",f1)
print("f1的类型是: ",type(f1))
print("f2:",f2)
print("f2的类型是: ",type(f2))
print("-"*10)

print("f3:",f3)
print("f3的类型是: ",type(f3))
```

```
print("f4:",f4)
print("f4的类型是: ",type(f4))
print("-"*10)

print("f5:",f5)
print("f5的类型是: ",type(f5))

'''
f1: 1.0
f1的类型是:  <class 'float'>
f2: -1.0
f2的类型是:  <class 'float'>
----------
f3: 0.0
f3的类型是:  <class 'float'>
f4: 1.0
f4的类型是:  <class 'float'>
----------
f5: 99.0
f5的类型是:  <class 'float'>
'''
```

str()

> 将int、float、bool、list、tuple、set、dict类型的数据转换为str类型

案例:

```
# 定义int类型变量
i1 = 1
# 定义float类型变量
f1 = 9.99
# 定义bool类型变量
b1 = False
b2 = True
# 定义list类型变量
l1 = [1, 2, 'a', 'b']
# 定义tuple类型变量
t1 = (1, 2, 'a', 'b')
# 定义set类型变量
s1 = {1, 2, 'a', 'b'}
# 定义dict类型变量
d1 = {'name': '小白', 'age':18}

# 使用str()函数
str1 = str(i1)
str2 = str(f1)
str3 = str(b1)
```

```
str4 = str(b2)
str5 = str(l1)
str6 = str(t1)
str7 = str(s1)
str8 = str(d1)

print("str1:",str1)
print("str1的类型是: ",type(str1))
print("-"*10)

print("str2:",str2)
print("str2的类型是: ",type(str2))
print("-"*10)

print("str3:",str3)
print("str3的类型是: ",type(str3))
print("str4:",str4)
print("str4的类型是: ",type(str4))
print("-"*10)

print("str5:",str5)
print("str5的类型是: ",type(str5))
print("-"*10)

print("str6:",str6)
print("str6的类型是: ",type(str6))
print("-"*10)

print("str7:",str7)
print("str7的类型是: ",type(str7))
print("-"*10)

print("str8:",str8)
print("str8的类型是: ",type(str8))

'''
str1: 1
str1的类型是:  <class 'str'>
----------
str2: 9.99
str2的类型是:  <class 'str'>
----------
str3: False
str3的类型是:  <class 'str'>
str4: True
str4的类型是:  <class 'str'>
----------
str5: [1, 2, 'a', 'b']
str5的类型是:  <class 'str'>
```

```
----------
str6: (1, 2, 'a', 'b')
str6的类型是:  <class 'str'>
----------
str7: {'b', 1, 2, 'a'}
str7的类型是:  <class 'str'>
----------
str8: {'name': '小白', 'age': 18}
str8的类型是:  <class 'str'>
'''
```

list()

> 将tuple、set、dict类型的数据转换为list类型。其中dict类型转换为list类型时，获取的列表中存储的值是dict类型变量的key值。

案例：

```
# 定义tuple变量
t1 = (1, 2, 'a', 'b')
# 定义set变量
s1 = {1, 2, 'a', 'b'}
# 定义dict变量
d1 = {'name': '小白', 'age':18}

# 使用list()函数
l1 = list(t1)
l2 = list(s1)
l3 = list(d1)

print("l1:",l1)
print("l1的类型是:",type(l1))
print("-"*10)

print("l2:",l2)
print("l2的类型是:",type(l2))
print("-"*10)

print("l3:",l3)
print("l3的类型是:",type(l3))

'''
l1: [1, 2, 'a', 'b']
l1的类型是:  <class 'list'>
----------
l2: [1, 2, 'b', 'a']
l2的类型是:  <class 'list'>
----------
```

```
l3: ['name', 'age']
l3的类型是:  <class 'list'>
'''
```

tuple()

> 将list、set、dict类型的数据转换为tuple类型。其中dict类型转换为tuple类型时获取的元祖中存储的值是dict类型变量的key值。

案例:

```
# 定义list变量
l1 = [1, 2, 'a', 'b']
# 定义set变量
s1 = {1, 2, 'a', 'b'}
# 定义dict变量
d1 = {'name': '小白', 'age':18}

# 使用tuple()函数
t1 = tuple(l1)
t2 = tuple(s1)
t3 = tuple(d1)

print("t1:",t1)
print("l1的类型是: ",type(t1))
print("-"*10)

print("t2:",t2)
print("t2的类型是: ",type(t2))
print("-"*10)

print("t3:",t3)
print("t3的类型是: ",type(t3))

'''
t1: (1, 2, 'a', 'b')
l1的类型是:  <class 'tuple'>
----------
t2: (1, 2, 'b', 'a')
t2的类型是:  <class 'tuple'>
----------
t3: ('name', 'age')
t3的类型是:  <class 'tuple'>
'''
```

set()

> 将list、tuple、dict类型的数据转换为set类型。其中dict类型转换为set类型时获取的元祖中存储的值是dict类型变量的key值。

案例:

```
# 定义list变量
l1 = [1, 2, 'a', 'b']
# 定义tuple变量
t1 = (1, 2, 'a', 'b')
# 定义dict变量
d1 = {'name': '小白', 'age':18}

# 使用set()函数
s1 = set(l1)
s2 = set(t1)
s3 = set(d1)

print("s1:",s1)
print("s1的类型是: ",type(s1))
print("-"*10)

print("s2:",s2)
print("s2的类型是: ",type(s2))
print("-"*10)

print("s3:",s3)
print("s3的类型是: ",type(s3))

'''
s1: {1, 2, 'b', 'a'}
s1的类型是:  <class 'set'>
----------
s2: {1, 2, 'b', 'a'}
s2的类型是:  <class 'set'>
----------
s3: {'age', 'name'}
s3的类型是:  <class 'set'>
'''
```

dict()

> 因为dict字典类型是键值对对应，所以list、tuple、set类型没法转换

```python
# 创建空的字典
d1 = dict()
print("d1:",d1)
print("d1的类型是:",type(d1))

'''
d1: {}
d1的类型是: <class 'dict'>
'''
```