第三章 PYTHON数据结构

序列(字符串)和集合

3.1字符串操作

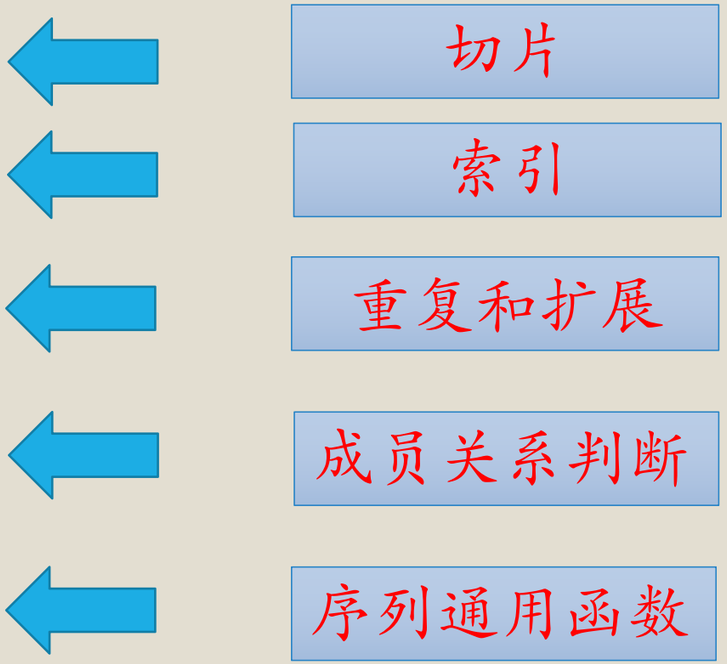
1：标准序列操作适用于字符串

- 索引，分片
- 乘法，加法
- 成员资格判断
- 求长度，最大值，最小值

2：字符串不可变：因此分片或单独赋值是不合法的

3.1 字符串操作

```
>>> x = '123456abcdef'
>>> x[1:8:3]
'25b'
>>> x[-5]#索引
'b'
>>> x*2+'xyz'
'123456abcdef123456abcdefxyz'
>>> '12' in x
True
>>> '13' in x
False
>>> len(x)
12
>>> max(x)
'f'
>>> min(x)
'1'
```



3.2字符串格式化

1：%格式化字符串

- 替代：用**元组内的元素**按照**预设格式**替代字符串中的内容。
- 转换：%s称为转换说明符，标记了需要插入的转换值的位置。s表示会被格式化为字符串—如果不是字符串会自动转换为字符串。

```
>>> x = "%s is %4.2fm tall, and student number is %3d"
>>> print(x % ("Lucy",1.623,12))
Lucy is 1.62m tall, and student number is  12
>>> print(x % ("Lily",1.637,131))
Lily is 1.64m tall, and student number is 131
>>> print( "Python is number %d" % 1)
Python is number 1
```

3.2字符串格式化

2: format方法与{ }替换字段

```
>>> x = '{}每天读{}本书，背{}个英文单词，晨跑{}米'
>>> x.format("李磊",2,12,800)
'李磊每天读2本书，背12个英文单词，晨跑800米'
>>> x.format("韩梅梅",1,20,1200)
'韩梅梅每天读1本书，背20个英文单词，晨跑1200米'
```

{}字段可以通过默认顺序进行替换：从左至由

3.2字符串格式化

2: format方法与{ }替换字段

```
>>> x = "今天是{}月{}日，东北风{}级，平均气温{}摄氏度"
>>> x = "今天是{3}月{2}日，东北风{0}级，平均气温{1}摄氏度"
>>> x.format(2,34,5,4)
'今天是4月5日，东北风2级，平均气温34摄氏度'
```

{}字段可以通过手工索引进行替换： 按照{}内的索引

3.2字符串格式化

2: format方法与{ }替换字段

```
>>> x = "今天是{year}年{month}月{day}日"
>>> x.format(year =2021, month = 9, day = 18)
'今天是2021年9月18日'
>>> x.format(month = 9, year =2021, day = 18)
'今天是2021年9月18日'
```

{}字段可以按照字段名替换

3.2字符串格式化

2: format方法与{ }替换字段

```
>>> x = "{name}的{1}考了{score}分，班级排名{0}"
>>> x.format(3, "英语", name = "李磊",score = 90)
'李磊的英语考了90分，班级排名3'
```

{}字段名和{}字段索引可以混用,位置参数在前，关键字参数在后

3.3 字符串基本转换

1：{:}内可填写字符串转换的格式

冒号前可写字段名或者索引，冒号后填写待格式化字符串的格式

```
>>> x = "{name:^10.8s}的{subject:<10s}成绩是{score:5.2f}分"
>>> x.format(name = "Danial", subject = "English", score = 75.534)
'  Danial  的English   成绩是75.53分'
```

3.3 字符串基本转换

2：格式说明

a) 格式符号：<表示左对齐， >表示右对齐，^表示居中；+表示转换值前加正负号，“ ”(空白字符)表示正数之前留空格；#表示展示数据进制类型前缀 ；0表示位数不够用0填充。

b) 最小字段宽度，转换后字符串至少应具有该值指定的宽度。

c) .后跟精度值：如果转换是实数，精度值就会出现在小数点后的位数，如果是字符串，那么该数值表示字符串的最大宽度。

d) 类型说明符（bcdeEfFgGosxX%）

3：类型说明符

符号	含义
b	表示将整数转换成二进制数
c	转化成Unicode码
d	整数默认设置，十进制
E(e)	科学计数法，用E(e)表示指数
F(f)	用浮点数表示，对于nan和inf用大（小）写表示
G(g)	自动在科学计数法和浮点数表示中做出选择，指数E(e)大小写取决于G(g)
o	将整数表示为八进制数
s	字符串表示
X(x)	十六进制表示
%	将数值表示为百分比值

3：类型说明符示例

```
>>> y = "{:^10.5s}的成绩是{:<+10.2f}分，班级排名第{:d}"
>>> y.format("李磊",95.5,5)
'  李磊  的成绩是+95.50  分，班级排名第5'
>>> y = "{num:+e}的二进制形式为{num:+#b},八进制为{num:+#o},十六进制为{num:+#X}"
>>> y.format(num=123)
'+1.230000e+02的二进制形式为+0b1111011,八进制为+0o173,十六进制为+0X7B'
>>> y.format(num=23)
'+2.300000e+01的二进制形式为+0b10111,八进制为+0o27,十六进制为+0X17'
>>> y = "{:5.3s}的年收入为{:5d}元,月平均{:5.2f}元"
>>> y.format("Danial",60023,60023/12)
'Dan  的年收入为60023元,月平均5001.92元'
>>> y = "{:5.3s}的年收入为{:4d}元,月平均{:5.2f}元"
>>> y.format("Danial",60023,60023/12)
'Dan  的年收入为60023元,月平均5001.92元'
>>> y.format("Tomy",67,67/12)
'Tom  的年收入为  67元,月平均 5.58元'
```

1：字符串的精度是对字符串的直接切片

2：浮点数精度存在四舍五入，首先保证小数部分，当长度超过宽度时，最初设置的宽度将不起作用；整数不允许使用精度

3：默认对齐方式，字符串左对齐，数值右对齐

4：理解下列代码含义

```
width = int(input('Please enter width: '))

price_width = 10
item_width  = width - price_width  #30

header_fmt = '{{:{}}}{{:>{}}}'.format(item_width, price_width)
#{{}}表示花括号，而非字符串中的替换字段
fmt        = '{{:{}}}{{:>{}.2f}}'.format(item_width, price_width)

print('=' * width)
print(header_fmt.format('Item', 'Price'))
print('-' * width)

print(fmt.format('Apples', 0.4))
print(fmt.format('Pears', 0.5))
print(fmt.format('Cantaloupes', 1.92))
print(fmt.format('Dried Apricots (16 oz.)', 8))
print(fmt.format('Prunes (4 lbs.)', 12))

print('=' * width)
```

字符串中{}表示替换字段，而花括号该如何表示?

3.4 字符串方法

```
>>> x = "What is your name?"
>>> x.endswith("?",3,8)
False
>>> x.endswith("?")#可以指定起点和终点（不包含终点）
True
>>> x.index("?") #能找到返回index
17
>>> x.index("?",8,12) #找不到出错
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    x.index("?",8,12)
ValueError: substring not found
>>> x.find("u",8,12)
10
>>> x.find("W",8,12) #能找到返回index，否则返回-1，注意和index区别
-1
>>> "What is your name?".count('a',2,10)#count 非常重要
1
```

find,index,count均可指定起点和终点

```
>>> "+".join('1234')
'1+2+3+4'
>>> "+".join(['1','2','3','4'])
'1+2+3+4'
>>> x = "+".join(['1','2','3','4'])
>>> x.split('+') #join, split互为相反的方法，非常重要
['1', '2', '3', '4']
>>> 'ABC'.lower()
'abc'
>>> 'abc'.upper()
'ABC'
>>> 'abc'.replace('b','')
'ac'
>>> 'abc'.replace('b','XXXXXXXXXXXXXXX')#replace 非常重要
'aXXXXXXXXXXXXXXXc'
>>>    What is your name ?    '.strip()
'What is your name ?'
```

```
>>> "What is your name?".translate(table)
'Wx1t vs your o1u5?'
>>> "What is your name?".title()
'What Is Your Name?'
>>> "What is your name?".istitle()
False
>>> '1234iaff'.isalpha()
False
>>> '1234iaff'.isalnum()
True
>>> '1234iaff'.isdecimal()
False
>>> '1234iaff'.isdigit()
False
>>> '123.4'.isdigit()
False
>>> '123.4'.isdecimal()
False
>>> '123.4'.isnumeric()
False
```

字符串方法极多，请后续查询和练习。

3.5 序列方法使用查询

```
>>> dir(list) #查询列表的方法
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__d
oc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__',
'__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le
__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__
', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__',
'__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']
>>> print(list.pop.__doc__)#打印出pop方法的文档
Remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.
```

类似地，可以查询字符串(str)，元组(tuple)，集合(set)和字典(dict)等内置数据类型的方法.

请每个同学查询字符串方法的说明文档，并练习使用这些方法。

3.6 string 库(常用字符集)

```
>>> import string
>>> dir(string)
['Formatter', 'Template', '_ChainMap', '_TemplateMetaclass', '__all__',
 '__builtins__', '__cached__', '__doc__', '__file__', '__loader__',
 '__name__', '__package__', '__spec__', '_re', '_string', 'ascii_letters',
 'ascii_lowercase', 'ascii_uppercase', 'capwords', 'digits', 'hexdigits',
 'octdigits', 'printable', 'punctuation', 'whitespace']
>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'
>>> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string.digits
'0123456789'
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

3.7 集合

集合与列表相似，都可以用来存储多个元素。不同于列表的是，集合中的元素彼此不能相同并且不按照任何特定的顺序放置。

可以将若干个元素用一对花括号 {} 括起来以创建一个集合。集合内的元素同样使用逗号分隔。一个集合也可以包含不同类型的元素。

Python使用内置类set来定义集合。使用set函数可以将列表、元组、字符串等类型转换为集合。

3.8 集合的基本操作

由于集合的元素是无序的，不能使用下标来访问集合中的元素。但是集合可以使用for...in循环来遍历其中的所有元素。使用in或not in运算符可以判断一个元素是否在一个集合中。

len函数同样适用于集合。可以使用len函数求集合的大小，使用max函数求集合最大元素，使用sum函数求集合内所有元素的和。

set类常用的成员函数如下:

- **set.add(elem)**
- **set.remove(elem)**
- **set.discard(elem)**
- **set.pop()**
- **set.clear()**

3.9 集合运算

Python提供了求交集、并集、差集和对称差集等集合运算。

- （1）使用s1.intersection(s2)或者s1&s2可以计算两个集合的交集。
- （2）使用s1.union(s2)或者s1 | s2可以计算两个集合的并集。
- （3）使用s1.difference(s2)或者s1-s2可以计算两个集合的差集。
- （4）使用s1.symmetric_difference(s2)或者s1^s2可以计算两个集合的对称差集。

3.10 集合方法查询与练习

```
>>> dir(set)
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__iter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference', 'difference_update', 'discard', 'intersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'pop', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union', 'update']
>>> print(set.difference.__doc__)
Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)
>>> {1,2,3}.difference({2,3,4})
{1}
```