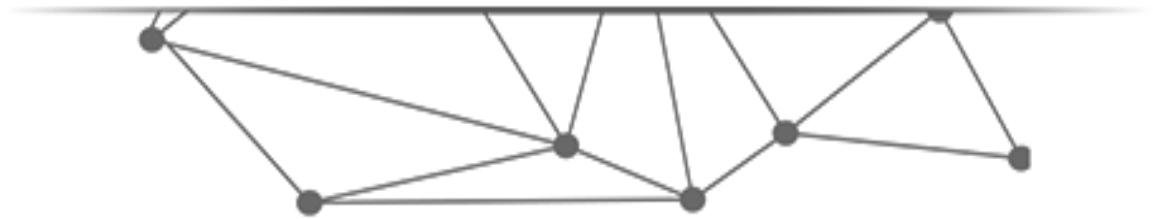
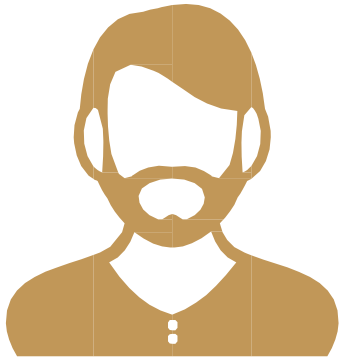
3.2 序列类型（字符串、列表、元组）



3.2序列类型及操作

- 序列类型定义
- 序列处理函数及方法
- 元组类型及操作
- 列表类型及操作
- 序列类型应用场景



序列类型定义

序列是具有先后关系的一组元素

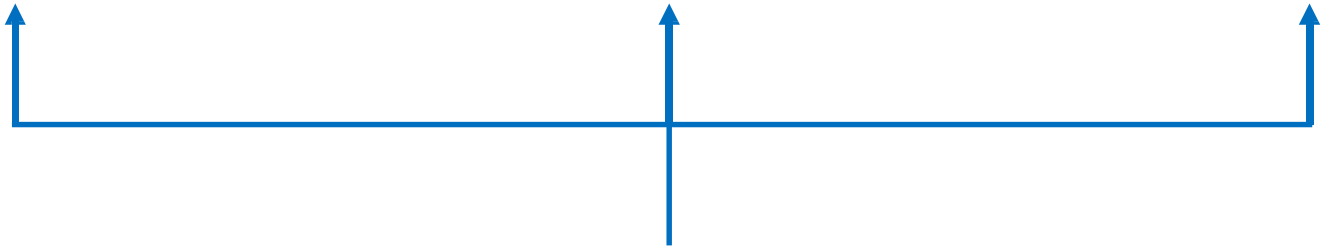
- **序列是一维元素向量，元素类型可以不同**
- **类似数学元素序列：** $s_0, s_1, \cdots, s_{n-1}$
- **元素间由序号引导，通过下标访问序列的特定元素**

python™

序列类型定义

序列是一个基类类型

字符串类型　　**元组类型**　　**列表类型**



序列类型

python™

序列类型定义

序号的定义

python™

字符串

由0个或多个字符组成的有序字符序列

- **字符串由一对单引号或一对双引号表示**

  "请输入带有符号的温度值:" **或者** 'C'

- **字符串是字符的有序序列，可以对其中的字符进行索引**

  "请" **是** "请输入带有符号的温度值:" **的第0个字符**

python™

字符串

字符串有 2类共4种 表示方法

- 由一对**单引号**或**双引号**表示，仅表示单行字符串

  "请输入带有符号的温度值:"　或者　'C'

- 由一对**三单引号**或**三双引号**表示，可表示多行字符串

  ''' Python

  语言 '''

Q: 老师老师，三引号不是多行注释吗?

Python语言为何提供 2类共4种 字符串表示方式?

字符串

字符串有 2类共4种 表示方法

- **如果希望在字符串中包含双引号或单引号呢?**

  '这里有个双引号(")' **或者** "这里有个单引号(')"

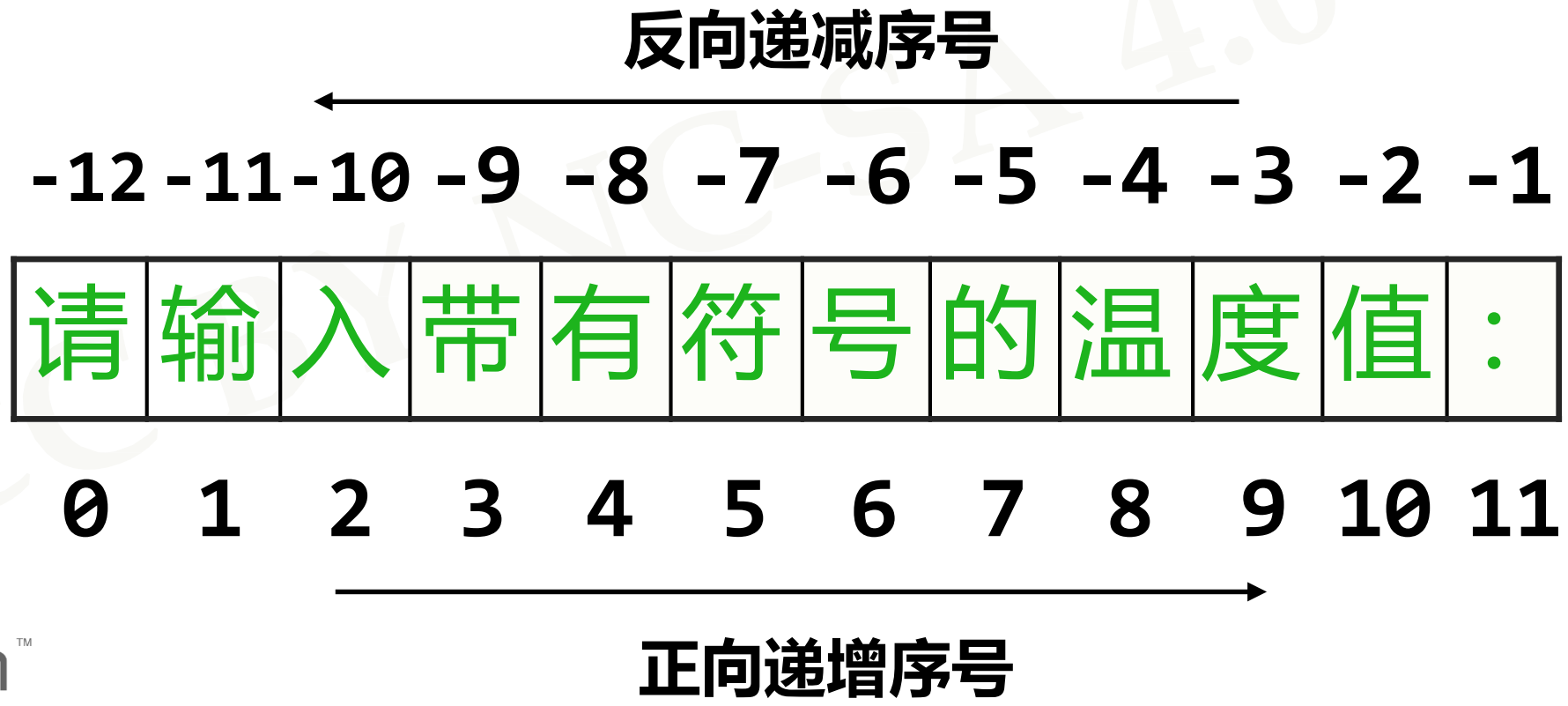
- **如果希望在字符串中既包括单引号又包括双引号呢?**

  ''' 这里既有单引号(')又有双引号 (") '''

python™

字符串的序号

正向递增序号 和 反向递减序号



字符串的使用

使用[ ]获取字符串中一个或多个字符

- **索引：返回字符串中单个字符　　<字符串>[M]**

```
"请输入带有符号的温度值: "[0]
```

- **切片：返回字符串中一段字符子串　　<字符串>[M: N]**

```
"请输入带有符号的温度值: "[1:3]
```

python™

字符串切片高级用法

使用[M: N: K]根据步长对字符串切片

- **<字符串>[M: N]，M缺失表示至开头，N缺失表示至结尾**

  "〇一二三四五六七八九十"[:3] **结果是** "〇一二"

- **<字符串>[M: N: K]，根据步长K对字符串切片**

  "〇一二三四五六七八九十"[1:8:2] **结果是** "一三五七"

  "〇一二三四五六七八九十"[::-1] **结果是** "十九八七六五四三二一〇"

字符串的特殊字符

转义符 \

- **转义符表达特定字符的本意**

  "这里有个双引号(\")" **结果为** 这里有个双引号(")

- **转义符形成一些组合，表达一些不可打印的含义**

  "\b"**回退** "\n"**换行(光标移动到下行首)** "\r" **回车(光标移动到本行首)**

python™

字符串操作符

由0个或多个字符组成的有序字符序列

操作符及使用	描述
x + y	连接两个字符串x和y
n * x 或 x * n	复制n次字符串x
x in s	如果x是s的子串，返回True，否则返回False

python™

字符串操作符

获取星期字符串

- 输入：1-7的整数，表示星期几
- 输出：输入整数对应的星期字符串
- 例如：输入3，输出 星期三

python™

字符串操作符

获取星期字符串

```
#WeekNamePrintV1.py
weekStr = "星期一星期二星期三星期四星期五星期六星期日"
weekId = eval(input("请输入星期数字(1-7): "))
pos = (weekId - 1 ) * 3
print(weekStr[pos: pos+3])
```

python™

字符串操作符

获取星期字符串

```
#WeekNamePrintV2.py
weekStr = "一二三四五六日"
weekId = eval(input("请输入星期数字(1-7): "))
print("星期" + weekStr[weekId-1])
```

python™

字符串处理函数

一些以函数形式提供的字符串处理功能

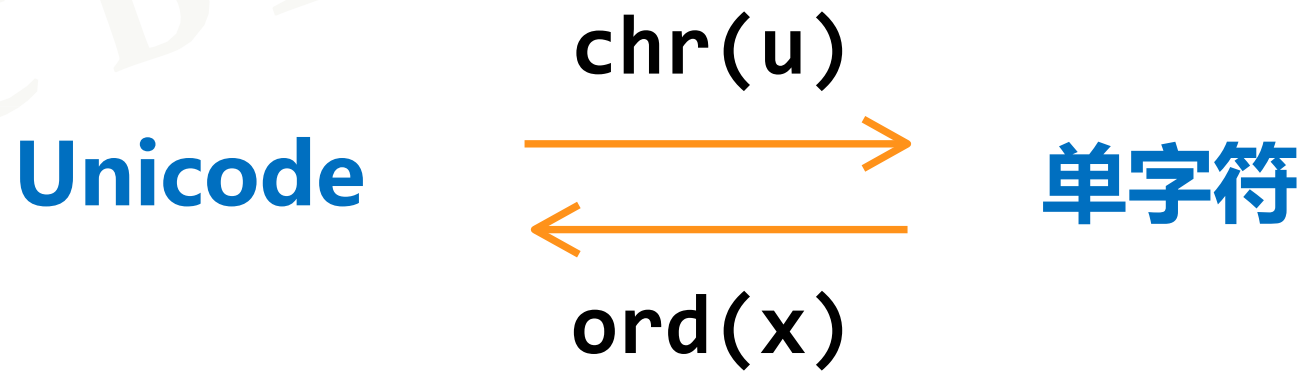
函数及使用	描述
len(x)	长度，返回字符串x的长度 len("一二三456") 结果为 6
str(x)	任意类型x所对应的字符串形式 str(1.23)结果为"1.23" str([1,2])结果为"[1,2]"
hex(x) **或** oct(x)	整数x的十六进制或八进制小写形式字符串 hex(425)结果为"0x1a9" oct(425)结果为"0o651"
chr(u)	x为Unicode编码，返回其对应的字符
ord(x)	x为字符，返回其对应的Unicode编码

python™

字符串处理函数

一些以函数形式提供的字符串处理功能

函数及使用	描述
chr(u)	x为Unicode编码，返回其对应的字符
ord(x)	x为字符，返回其对应的Unicode编码



python™

Unicode编码

Python字符串的编码方式

- 统一字符编码，即覆盖几乎所有字符的编码方式
- 从0到1114111 (0x10FFFF)空间，每个编码对应一个字符
- Python字符串中每个字符都是Unicode编码字符

python™

Unicode编码

一些有趣的例子（改成凯撒密码）

```
>>> "1 + 1 = 2 " + chr(10004)
'1 + 1 = 2 ✓'
>>> "这个字符♉的Unicode值是： " + str(ord("♉"))
'这个字符♉的Unicode值是：  9801'
>>> for i in range(12):
        print(chr(9800 + i), end="")
♈♉♊♋♌♍♎♏♐♑♒♓
```

挑战题

恺撒密码是古罗马凯撒大帝用来对军事情报进行加解密的算法，它采用了替换方法对信息中的每一个英文字符循环替换为字母表序列中该字符后面的第三个字符，即，字母表的对应关系如下：

原文：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

密文：D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

对于原文字符P，其密文字符C满足如下条件：C = (P+3) mod 26

上述是凯撒密码的加密方法，解密方法反之，即：P = (C-3) mod 26

假设用户可能使用的输入仅包含西文字母，即英文大小写字母a~zA~Z和特殊字符，请编写一个程序，对输入字符串进行凯撒密码加密，直接输出结果，其中特殊字符不进行加密处理。

注意要点：

需要区分英文字母的大小写。

python™

字符串处理方法

"方法"在编程中是一个专有名词

- **"方法"特指<a>.<b>()风格中的函数<b>()**
- **方法本身也是函数，但与<a>有关，<a>.<b>()风格使用**
- **字符串或字符串变量是<a>，存在一些可用方法**

python™

字符串处理方法

一些以方法形式提供的字符串处理功能

方法及使用	描述
str.lower() **或** str.upper()	返回字符串的副本，全部字符小写/大写"AbCdEfGh".lower() 结果为 "abcdefgh"
str.split(sep=None)	返回一个列表，由str根据sep被分隔的部分组成"A,B,C".split(",") 结果为: ['A','B','C']
str.count(sub)	返回子串sub在str中出现的次数"an apple a day".count("a") 结果为: 4
str.replace(old, new)	返回字符串str副本，所有old子串被替换为new"python".replace("n","n123.io") 结果为: "python123.io"
str.center(width[,fillchar])	字符串str根据宽度width居中，fillchar可选。"python".center(20,"=") 结果为: '=======python======='
str.strip(chars)	从str中去掉在其左侧和右侧chars中列出的字符。"= python= ".strip("=np") 结果为: "ytho"
str.join(iter)	在iter变量除最后元素外每个元素后增加一个str。",".join("12345") 结果为: "1,2,3,4,5" #主要用于字符串分隔等

python™

字符串类型的格式化

格式化是对字符串进行格式表达的方式

- **字符串格式化使用.format()方法，用法如下：**

<模板字符串>.format(<逗号分隔的参数>)

python™

字符串类型的格式化

槽

```
"{ }:计算机{ }的CPU占用率为{ }%".format("2018-10-10","C",10)
```

0 1 2

0 1 2

字符串中槽{}的默认顺序

format()中参数的顺序

python™

字符串类型的格式化

槽

```
"{1}:计算机{0}的CPU占用率为{2}%".format("2018-10-10","C",10)
```


python™

序列处理函数及方法

序列类型通用操作符　6个

操作符及应用	描述
x in s	如果x是序列s的元素，返回True，否则返回False
x not in s	如果x是序列s的元素，返回False，否则返回True
s + t	连接两个序列s和t
s*n 或 n*s	将序列s复制n次
s[i]	索引，返回s中的第i个元素，i是序列的序号
s[i: j] 或 s[i: j: k]	切片，返回序列s中第i到j以k为步长的元素子序列

python™

序列类型操作实例

```
>>> ls = ["python", 123, ".io"]
>>> ls[::-1]
['.io', 123, 'python']
>>> s = "python123.io"
>>> s[::-1]
'oi.321nohtyp'
```

PY01B27 兼儒墨

python™

序列类型通用函数和方法

5个函数和方法

函数和方法	描述
len(s)	返回序列s的长度，即元素个数
min(s)	返回序列s的最小元素，s中元素需要可比较
max(s)	返回序列s的最大元素，s中元素需要可比较
s.index(x) 或 s.index(x, i, j)	返回序列s从i开始到j位置中第一次出现元素x的位置
s.count(x)	返回序列s中出现x的总次数

python™

序列类型操作实例

```
>>> ls = ["python", 123, ".io"]
>>> len(ls)
3
>>> s = "python123.io"
>>> max(s)
'y'
```

python™

列表类型定义

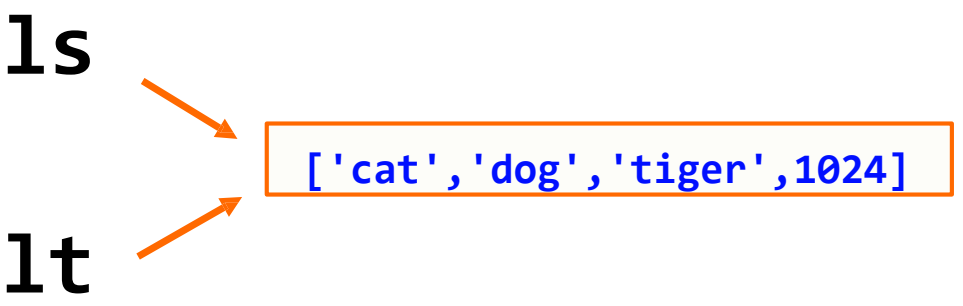
列表是序列类型的一种扩展，十分常用

- **列表是一种序列类型，创建后可以随意被修改**
- **使用方括号 [] 或list() 创建，元素间用逗号，分隔**
- **列表中各元素类型可以不同，无长度限制**

python™

列表类型定义

```
>>> ls = ["cat", "dog", "tiger", 1024]
>>> ls
['cat', 'dog', 'tiger', 1024]
>>> lt = ls
>>> lt
['cat', 'dog', 'tiger', 1024]
```



方括号 [] 真正创建一个列表，赋值仅传递引用

列表类型操作函数和方法

函数或方法	描述
ls[i] = x	替换列表ls第i元素为x
ls[i: j: k] = lt	用列表lt替换ls切片后所对应元素子列表
del ls[i]	删除列表ls中第i元素
del ls[i: j: k]	删除列表ls中第i到第j以k为步长的元素
ls += lt	更新列表ls，将列表lt元素增加到列表ls中
ls *= n	更新列表ls，其元素重复n次

python™

列表类型操作

```
>>> ls = ["cat", "dog", "tiger", 1024]
>>> ls[1:2] = [1, 2, 3, 4]
['cat', 1, 2, 3, 4, 'tiger', 1024]
>>> del ls[::3]
[1, 2, 4, 'tiger']
>>> ls*2
[1, 2, 4, 'tiger', 1, 2, 4, 'tiger']
```

python™

列表类型操作函数和方法

函数或方法	描述
ls.append(x)	在列表ls最后增加一个元素x
ls.clear()	删除列表ls中所有元素
ls.copy()	生成一个新列表，赋值ls中所有元素
ls.insert(i,x)	在列表ls的第i位置增加元素x
ls.pop(i)	将列表ls中第i位置元素取出并删除该元素
ls.remove(x)	将列表ls中出现的第一个元素x删除
ls.reverse()	将列表ls中的元素反转

python™

列表类型操作

```
>>> ls = ["cat", "dog", "tiger", 1024]
>>> ls.append(1234)
['cat', 'dog', 'tiger', 1024, 1234]
>>> ls.insert(3, "human")
['cat', 'dog', 'tiger', 'human', 1024, 1234]
>>> ls.reverse()
[1234, 1024, 'human', 'tiger', 'dog', 'cat']
```

python™

列表功能默写

- 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
- 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

python™

列表功能默写

- 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]
```

python™

列表功能默写

```
>>> 0 in lt
>>> lt.append(0)
>>> lt.index(0)
>>> len(lt)
>>> max(lt)
>>> lt.clear()
```

- 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

python™

元组类型定义

元组是序列类型的一种扩展

- 元组是一种序列类型，一旦创建就不能被修改
- 使用小括号 ( ) 或 tuple() 创建，元素间用逗号 , 分隔
- 可以使用或不使用小括号

```
def func():
    return 1,2
```

python™

元组类型定义

```
>>> creature = "cat", "dog", "tiger", "human"
>>> creature
('cat', 'dog', 'tiger', 'human')
>>> color = (0x001100, "blue", creature)
>>> color
(4352, 'blue', ('cat', 'dog', 'tiger', 'human'))
```

python™

元组类型操作

元组继承序列类型的全部通用操作

- 元组继承了序列类型的全部通用操作
- 元组因为创建后不能修改，因此没有特殊操作
- 使用或不使用小括号

python™

元组类型操作

```
>>> creature = "cat", "dog", "tiger", "human"
>>> creature[::-1]
('human', 'tiger', 'dog', 'cat')
>>> color = (0x001100, "blue", creature)
>>> color[-1][2]
'tiger'
```

python™

序列类型应用场景

python™

序列类型应用场景

数据表示：元组 和 列表

- 元组用于元素不改变的应用场景，更多用于固定搭配场景
- 列表更加灵活，它是最常用的序列类型
- 最主要作用：表示一组有序数据，进而操作它们

python™

序列类型应用场景

元素遍历

```
for item in ls :
    <语句块>
```

```
for item in tp :
    <语句块>
```

python™

序列类型应用场景

数据保护

- 如果不希望数据被程序所改变，转换成元组类型

```
>>> ls = ["cat", "dog", "tiger", 1024]
>>> lt = tuple(ls)
>>> lt
('cat', 'dog', 'tiger', 1024)
```

python

序列类型及操作

- 序列是基类类型，扩展类型包括：字符串、元组和列表
- 元组用()和tuple()创建，列表用[]和set()创建
- 元组操作与序列操作基本相同
- 列表操作在序列操作基础上，增加了更多的灵活性

python™