上次课回顾：函数与代码复用

- 什么是函数：
  - 具有**名字**的**子程序**
  - 能接受参数
  - 返回输出

- 为什么要有函数：
  - 从程序设计语言的角度：
    - 支持代码复用
    - 提供对过程的封装和抽象
  - 从设计程序的角度：
    - 问题抽象和分解，降低编程难度

上次课回顾：函数与代码复用

- 定义函数:
  - 使用def语句：

    def <函数名>(<参数列表>):
        <函数体>
        return <返回值列表>
  - 使用lambda表达式：

    <函数名> = lambda <参数列表>: <返回值表达式>
    - 等价于：def <函数名>(<参数列表>):
        return <返回值表达式>
  - 实现原理：创建了一个function类型的**对象**，再将其**赋值**给名称为<函数名>的变量

上次课回顾：函数与代码复用

- 调用函数：

  最常见的调用方法：<函数名>(<实际传递的参数列表>)

  实现原理: 表达式求值!

  ()是一个优先级高很高的运算符，其含义是把左边的值**当做函数来调用**（？？？）

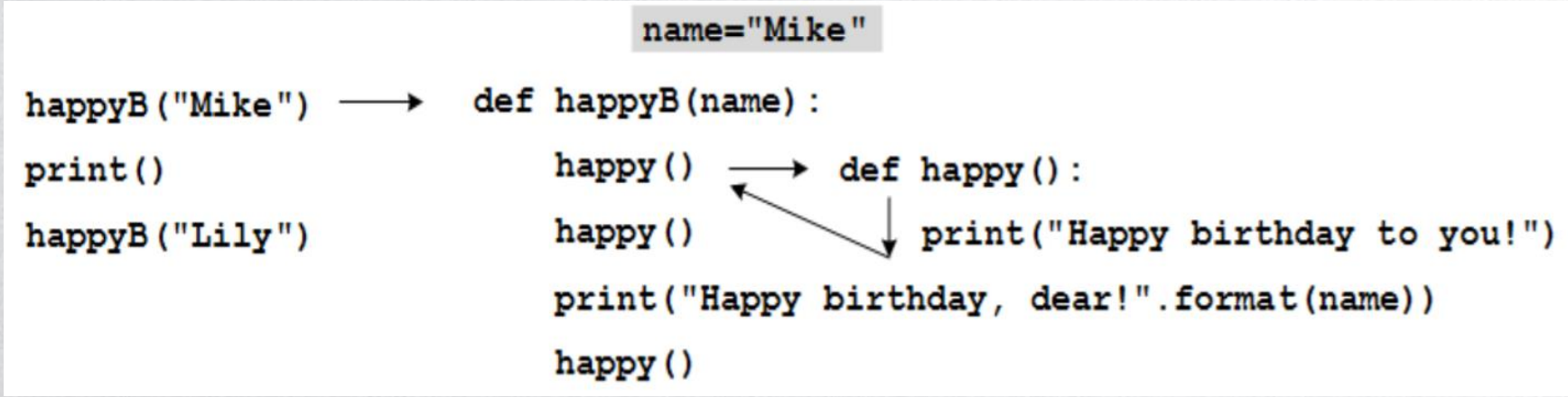
```
In [4]:  1 (lambda a, b: str(a)+str(b))(12,34)
Out[4]: '1234'

In [5]:  1 f = str
         2 f(123)
Out[5]: '123'

In [6]:  1 1.0("abc")
```

```
<>:1: SyntaxWarning: 'float' object is not callable; perhaps you missed a comma?
<>:1: SyntaxWarning: 'float' object is not callable; perhaps you missed a comma?
/var/folders/4m/kw1nckdn0qzcz68fpxyc8yj80000gn/T/ipykernel_58984/4135447890.py:1: SyntaxWarning: 'float' object is no
t callable; perhaps you missed a comma?
  1.0("abc")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/4m/kw1nckdn0qzcz68fpxyc8yj80000gn/T/ipykernel_58984/4135447890.py in <module>
----> 1 1.0("abc")

TypeError: 'float' object is not callable
```

上次课回顾：函数与代码复用

- 四个步骤：
  （1）调用程序在调用处暂停执行
  （2）在调用时将实参复制（赋值！）给函数的形参
  （3）执行函数体语句
  （4）函数调用结束给出返回值，程序回到调用前的暂停处继续执行



上次课回顾：函数与代码复用

- 函数参数传递：
  - 可选参数：
    - 在定义函数时，有些参数可以存在默认值
    - 函数被调用时，如果没有传入对应的参数值，则使用函数定义时的默认值代替
    - 可选参数必须定义在非可选参数后面
  - 可变数量参数：
    - 在函数定义时，可以设计可变数量参数
    - 可变参数被当作**元组（tuple）**类型传入函数中

```
In [6]: def dup(str, times=2):
            print(str*times)
        dup("ruc~")

ruc~ruc~

In [7]: dup("ruc!",4)

ruc!ruc!ruc!ruc!
```

```
In [9]: def vfunc(a, *b):
            print(type(b))
            for n in b:
                a += n
            return a
        vfunc(1,2,3,4,5)

<class 'tuple'>

Out[9]: 15
```

上次课回顾：函数与代码复用

- 函数的返回值：
  - return语句用来退出函数并将程序返回到函数被调用的位置继续执行
  - return语句同时可以将0个、1个或多个函数运算完的结果返回给函数被调用处的变量
    - 用return返回多个值，多个值以**元组**类型保存
  - 函数可以没有return，此时函数会在执行完函数体所有语句后，返回None

```
In [10]: def func(a,b):
             return a*b
         s = func("ruc!",2)
         print(s)

         ruc!ruc!
```

```
In [11]: def func(a,b):
             return b, a
         s = func("ruc!",2)
         print(s, type(s))

         (2, 'ruc!') <class 'tuple'>
```

上次课回顾：函数与代码复用

- 函数对变量的作用：局部变量 vs. 全局变量
  - 实现封装：函数内部使用局部变量不会影响全局变量，也可以不受全局变量的影响
  - 实现原理：Python解释器会按照如下规则找到变量对应的值
    - 首先在局部变量中查找
    - 若找不到，则到函数定义时的上一级代码块中查找
      - 最后会找到在没有缩进的代码块中定义的全局变量
    - 若在函数内对变量进行赋值操作，则默认该变量是一个局部变量
    - 使用global语句，可以改变赋值操作的默认行为，不生成局部变量
      - 这样在函数内部就可以通过赋值语句更改全局变量的值了
    - 产生新问题：一个变量可以既是全局变量又是局部变量？
      - Python禁止了这种行为，抛出UnboundLocalError异常

上次课回顾：函数与代码复用

- 区分函数定义和函数调用：
  - 函数定义时：
    - 编译(?)函数体中的语句，构建函数类型的对象
    - 把函数对象赋值给函数名指定的变量
    - 但并不会立即执行这些语句
    - 所以函数体中的语句可以包含当时未被定义的变量名
  - 函数被调用时：
    - 函数体中的语句被执行
    - Python解释器按照上页PPT中的方式去查找变量对应的值，进行表达式求值、函数调用……
    - 注意：这时函数本身在函数被定义的代码块中已经被定义了
      - 因此，函数内部可以调用函数自身！

```
1 from inspect import dis
2 def f(a, b):
3     c = a + b
4     return c
5 dis.dis(f)
```

```
3           0 LOAD_FAST                0 (a)
            2 LOAD_FAST                1 (b)
            4 BINARY_ADD
            6 STORE_FAST               2 (c)

4           8 LOAD_FAST                2 (c)
           10 RETURN_VALUE
```

```
def f1():
    f2()
def f2():
    print("This is function f2()")
f1()
```

递归的定义

- 函数作为一种代码封装，可以被其他程序调用，当然，也可以被函数内部代码调用。这种函数定义中调用函数自身的方式称为递归。
- 递归在数学和计算机应用上非常强大，能够非常简洁的解决重要问题。
- 数学上有个经典的递归例子叫阶乘，阶乘通常定义为：

$$n! = n(n-1)(n-2)...(1)$$

$$n! = \begin{cases} 1 & n = 0 \\ n(n-1)! & otherwise \end{cases}$$

- 递归的两个关键特征：
  - 存在一个或多个基例，基例不需要再次递归，它是确定的表达式；
  - 所有递归链要以一个或多个基例结尾。

递归的使用方法

- 阶乘计算：用户输入整数n，计算并输出n的阶乘值

In [2]:
```
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n-1)
num = eval(input("请输入一个整数: "))
print(fact(abs(int(num))))
```

```
请输入一个整数: 10
3628800
```

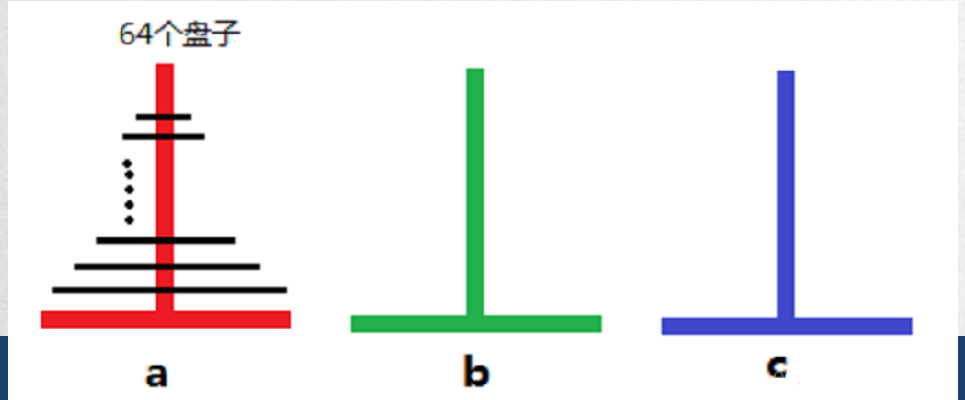
Python内置函数

- Python解释器提供了68个内置函数

abs()	id()	round()	compile()	locals()
all()	input()	set()	dir()	map()
any()	int()	sorted()	exec()	memoryview()
asci()	len()	str()	enumerate()	next()
bin()	list()	tuple()	filter()	object()
bool()	max()	type()	format()	property()
chr()	min()	zip()	frozenset()	repr()
complex()	oct()		getattr()	setattr()
dict()	open()		globals()	slice()
divmod()	ord()	bytes()	hasattr()	staticmethod()
eval()	pow()	delattr()	help()	sum()
float()	print()	bytearray()	isinstance()	super()
hash()	range()	callable()	issubclass()	vars()
hex()	reversed()	classmethod()	iter()	__import()__

课后练习2（请在未来课堂系统提交）

- 汉诺塔问题
  - 汉诺塔(Tower of Hanoi)源于印度传说中，大梵天创造世界时造了三根金钢石柱子，其中一根柱子自底向上叠着64片黄金圆盘。大梵天命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。
  - 请用Python编写一个汉诺塔的移动函数，将所有圆盘从A移动到C，一次只能移动一个盘子，盘子只能在3个标杆之间移动，更大的盘子不能放在更小的盘子上面。要求输入汉诺塔的层数（有多少个盘子），输出整个移动流程。



《人工智能与Python程序设计》——组合数据类型1

人工智能与Python程序设计 教研组

Python AI
组合数据类型

提纲

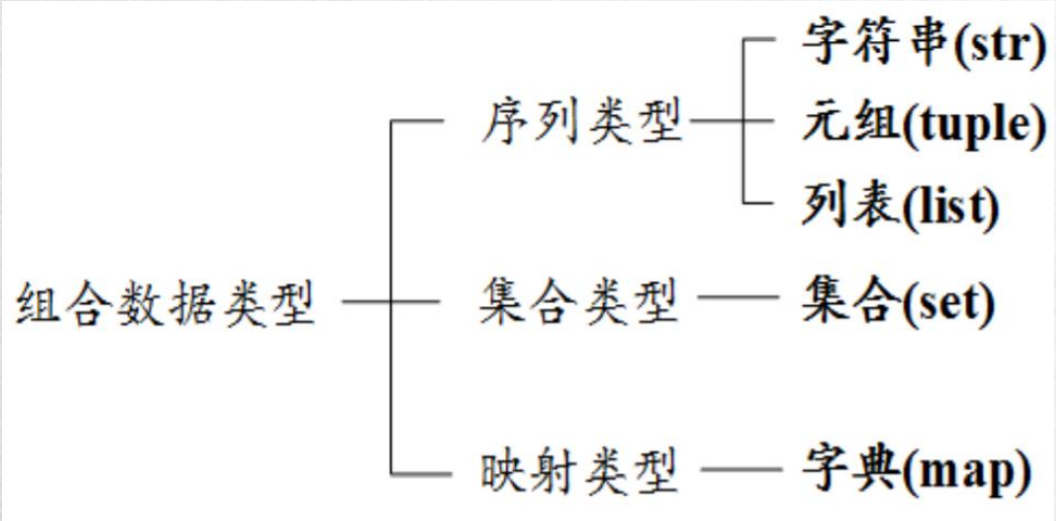
- 组合数据类型概述
- 列表类型与操作
- jieba库的使用与词频统计

组合数据类型概述

- 计算机不仅对单个变量表示的数据进行处理，更多情况，计算机需要对**一组数据**进行批量处理。一些例子包括：
  - 给定一组单词{python, data, function, list, loop}，计算并输出每个单词的长度；
  - 给定一个学院学生信息，统计一下男女生比例；
  - 一次实验产生了很多组数据，对这些大量数据进行分析；
- 基本数据类型仅能表示一个数据
- 实际计算中存在大量同时处理多个数据的情况，这需要将多个数据有效组织起来并统一表示，这种能够表示多个数据的类型称为**组合数据类型**

组合数据类型概述

- 组合数据类型能够将多个同类型或不同类型的数据组织起来，通过单一的表示使数据操作更有序更容易。
- 根据数据之间的关系，组合数据类型可以分为三类：序列类型、集合类型和映射类型。



序列类型

- 序列类型是一维元素向量，元素之间存在先后关系，通过序号访问
- 当需要访问序列中某特定值时，只需要通过下标标出即可
- 由于元素之间存在顺序关系，所以序列中可以存在相同数值但位置不同的元素。序列类型支持成员关系操作符（in）、长度计算函数（len()）、分片（[]），元素本身也可以是序列类型。

序列类型

- Python语言中有很多数据类型都是序列类型，其中比较重要的是：str（字符串）、tuple（元组）、list（列表）
  - 元组是包含0个或多个数据项的不可变序列类型。元组生成后是固定的，其中任何数据项不能替换或删除。
  - 列表则是一个**可以修改**数据项的序列类型，使用也最灵活。
- 索引体系：

序列类型

- 序列类型通用操作符和函数

操作符	描述
x in s	如果x是s的元素，返回True，否则返回False
x not in s	如果x不是s的元素，返回True，否则返回False
s + t	连接s和t
s * n 或 n * s	将序列s复制n次
s[i]	索引，返回序列的第i个元素
s[i: j]	分片，返回包含序列s第i到j个元素的子序列（不包含第j个元素）
s[i: j: k]	步骤分片，返回包含序列s第i到j个元素以j为步数的子序列
len(s)	序列s的元素个数（长度）
min(s)	序列s中的最小元素
max(s)	序列s中的最大元素
s.index(x[, i[, j]])	序列s中从i开始到j位置中第一次出现元素x的位置
s.count(x)	序列s中出现x的总次数

元组

- 元组（tuple）是序列类型中比较特殊的类型，因为它**一旦创建就不能被修改**。元组类型在表达固定数据项、函数多返回值、多变量同步赋值、循环遍历等情况下十分有用。
- Python中元组采用逗号和圆括号（可选）来表示。

```
In [1]: creature = "cat", "dog", "tiger", "human"
        creature
Out[1]: ('cat', 'dog', 'tiger', 'human')

In [2]: color = ("red", 0x001100, "blue", creature)
        color
Out[2]: ('red', 4352, 'blue', ('cat', 'dog', 'tiger', 'human'))

In [3]: color[2]
Out[3]: 'blue'

In [4]: color[-1][2]
Out[4]: 'tiger'
```

```
In [5]: def func(x): #函数多返回值
            return x, x**3
        a, b = 'dog', 'tiger' #多变量同步赋值
        a, b = (b, a) #多变量同步赋值，括号可省略
        print(a)

tiger

In [6]: import math
        for x, y in ((1,0), (2,5), (3,8)): #循环遍历
            print(math.hypot(x,y)) #求多个坐标值到原点的距离

1.0
5.385164807134505
8.54400374531753
```

集合类型

- 集合类型与数学中集合的概念一致，即包含0个或多个数据项的无序组合。集合中元素不可重复，元素类型只能是固定数据类型，例如：整数、浮点数、字符串、元组等。列表、字典和集合类型本身都是可变数据类型，不能作为集合的元素出现。
- 由于集合是**无序组合**，它没有索引和位置的概念，不能分片，集合中元素可以动态增加或删除(**可以修改**)。集合用大括号（{}）表示，可以用赋值语句生成一个集合。

集合类型

- 由于集合元素是无序的，集合的打印效果与定义顺序可以不一致。由于集合元素独一无二，使用集合类型能够过滤掉重复元素。set(x)函数可以用于生成集合。

```
In [7]: S = {"信息楼", "RUC", (12, "AI"), 100872}
        S
Out[7]: {(12, 'AI'), 100872, 'RUC', '信息楼'}

In [8]: S = {"信息楼", "RUC", (12, "AI"), 100872, "RUC"}
        S
Out[8]: {(12, 'AI'), 100872, 'RUC', '信息楼'}

In [10]: T = set('intelligence')
         T
Out[10]: {'c', 'e', 'g', 'i', 'l', 'n', 't'}

In [11]: T = set(("python", "learning", "artificial", "intelligence"))
         T
Out[11]: {'artificial', 'intelligence', 'learning', 'python'}
```

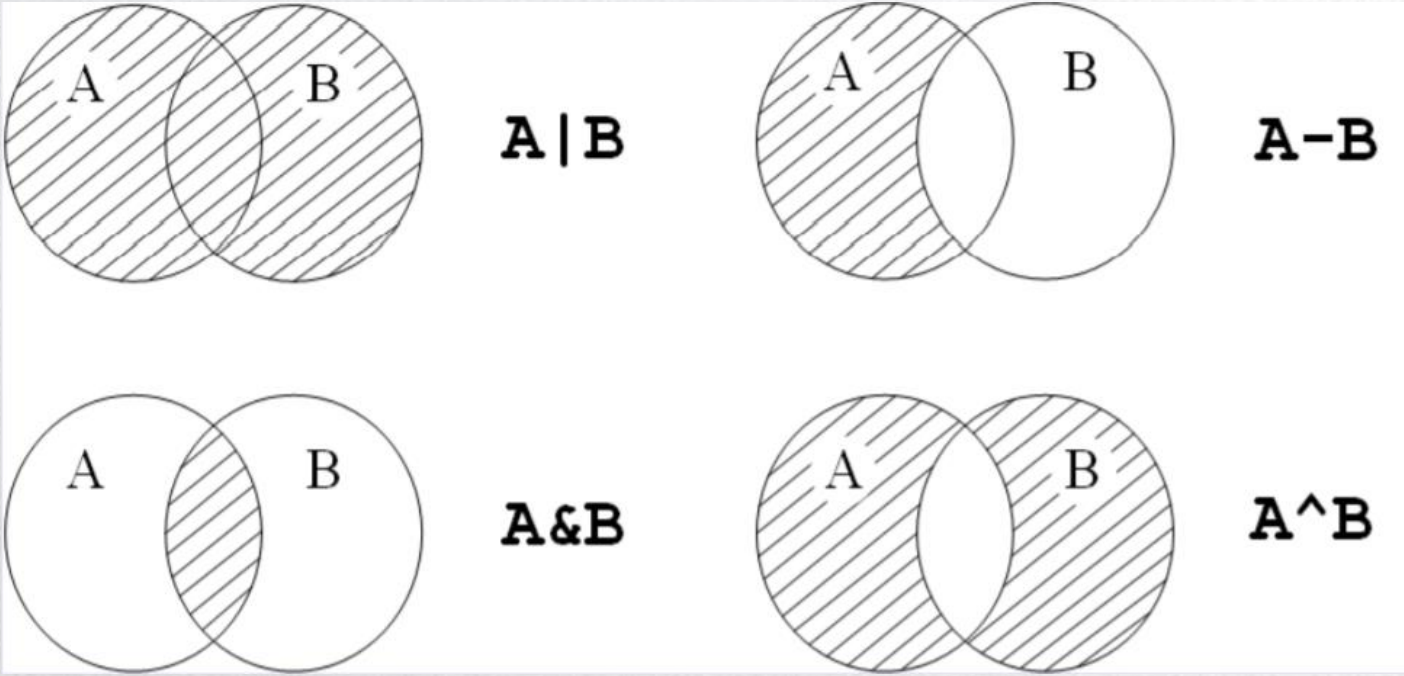
集合类型

- 集合操作符

操作符	描述
S – T 或 S.difference(T)	返回一个新集合，包括在集合S中但不在集合T中的元素
S-=T或S.difference_update(T)	更新集合S，包括在集合S中但不在集合T中的元素
S & T或S.intersection(T)	返回一个新集合，包括同时在集合S和T中的元素
S&=T或S.intersection_update(T)	更新集合S，包括同时在集合S和T中的元素。
S^T或s.symmetric_difference(T)	返回一个新集合，包括集合S和T中元素，但不包括同时在其中的元素
S=^T或s.symmetric_difference_update(T)	更新集合S，包括集合S和T中元素，但不包括同时在其中的元素
S|T或S.union(T)	返回一个新集合，包括集合S和T中所有元素
S=|T或S.update(T)	更新集合S，包括集合S和T中所有元素
S<=T或S.issubset(T)	如果S与T相同或S是T的子集，返回True，否则返回False，可以用S<T判断S是否是T的真子集
S>=T或S.issuperset(T)	如果S与T相同或S是T的超集，返回True，否则返回False，可以用S>T判断S是否是T的真超集

集合类型

- 上述操作符表达了集合类型的4种基本操作，交集（&）、并集（|）、差集（-）、补集（^），操作逻辑与数学定义相同



集合类型

- 集合类型有10个操作函数或方法

函数或方法	描述
S.add(x)	如果数据项x不在集合S中，将x增加到s
S.clear()	移除S中所有数据项
S.copy()	返回集合S的一个拷贝
S.pop()	随机返回集合S中的一个元素，如果S为空，产生KeyError异常
S.discard(x)	如果x在集合S中，移除该元素；如果x不在，不报错
S.remove(x)	如果x在集合S中，移除该元素；不在产生KeyError异常
S.isdisjoint(T)	如果集合S与T没有相同元素，返回True
len(S)	返回集合S元素个数
x in S	如果x是S的元素，返回True，否则返回False
x not in S	如果x不是S的元素，返回True，否则返回False

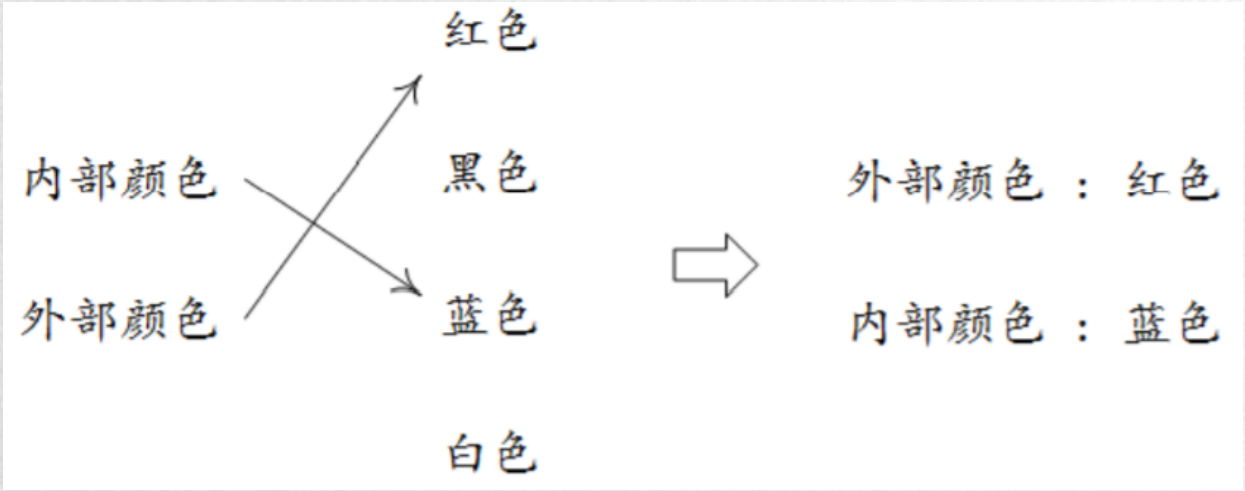
集合类型

- 集合类型主要用于三个场景：成员关系测试、元素去重和删除数据项。
- 集合类型与其他类型最大的不同在于它不包含重复元素，因此，当需要对一维数据进行去重或进行数据重复处理时，一般通过集合来完成。

```
In  [12]: S = {"信息楼", "RUC", (12, "AI"), 100872}
          "RUC" in S   #成员关系测试
Out[12]: True

In  [13]: tup = (125, "信息楼", "RUC", (12, "AI"), 100872, 125)
          set(tup)  #元素去重
Out[13]: {(12, 'AI'), 100872, 125, 'RUC', '信息楼'}

In  [29]: S = set(tup)-{125}
          S
Out[29]: {(12, 'AI'), 100872, 'RUC', '信息楼'}

In  [26]: newtup = tuple(S)  # 去重同时删除数据项
          newtup
Out[26]: (100872, '信息楼', (12, 'AI'), 'RUC')

In  [33]: S = set(tup)-{'信息楼'}
          S
Out[33]: {(12, 'AI'), 100872, 125, 'RUC'}
```

映射类型

- 映射类型是“键-值”数据项的组合，每个元素是一个键值对，即元素是(key, value)，元素之间是无序的。键值对(key, value)是一种二元关系。在Python中，映射类型主要以字典（dict）体现。



Python AI
组合数据类型

提纲

- 组合数据类型概述
- **列表类型与操作**
- jieba库的使用与词频统计

列表类型

- 列表（list）是包含0个或多个对象引用的有序序列，属于序列类型。与元组不同，列表的长度和内容都是可变的，可自由对列表中数据项进行增加、删除或替换。列表没有长度限制，元素类型可以不同，使用非常灵活。
- 由于列表属于序列类型，所以列表也支持成员关系操作符（in）、长度计算函数（len()）、分片（[]）。列表可以同时使用正向递增序号和反向递减序号，可以采用标准的比较操作符（<、<=、==、!=、>=、>）进行比较，列表的比较实际上是单个数据项的逐个比较。

列表类型

- 列表用中括号（[]）表示，也可以通过list()函数将元组或字符串转化成列表。直接使用list()函数会返回一个空列表。

```
In [35]: ls = ["信息楼", "RUC", [125, "AI"], 100872, "RUC"]
         ls
Out[35]: ['信息楼', 'RUC', [125, 'AI'], 100872, 'RUC']

In [36]: ls[2][-1][0]
Out[36]: 'A'

In [37]: list(("信息楼", "RUC", (12, "AI"), 100872, "RUC"))
Out[37]: ['信息楼', 'RUC', (12, 'AI'), 100872, 'RUC']

In [38]: list("中国人民大学高瓴人工智能学院")
Out[38]: ['中', '国', '人', '民', '大', '学', '高', '瓴', '人', '工', '智', '能', '学', '院']

In [39]: list()
Out[39]: []
```

列表类型

- 列表必须通过显式的数据赋值才能生成，简单将一个列表赋值给另一个列表不会生成（?)新的列表对象。

```
In [40]: ls = ["信息楼", "RUC", [125, "AI"], 100872, "RUC"]
         lt = ls
         ls[2] = 0
         lt
Out[40]: ['信息楼', 'RUC', 0, 100872, 'RUC']
```

列表类型

- 列表类型的操作

函数或方法	描述
ls[i] = x	替换列表ls第i数据项为x
ls[i: j] = lt	用列表lt替换列表ls中第i到j项数据（不含第j项，下同）
ls[i: j: k] = lt	用列表lt替换列表ls中第i到j以k为步的数据
del ls[i: j]	删除列表ls第i到j项数据，等价于ls[i: j]=[]
del ls[i: j: k]	删除列表ls第i到j以k为步的数据
ls += lt或ls.extend(lt)	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
ls.append(x)	在列表ls最后增加一个元素x
ls.clear()	删除ls中所有元素
ls.copy()	生成一个新列表，复制ls中所有元素
ls.insert(i, x)	在列表ls第i位置增加元素x
ls.pop(i)	将列表ls中第i项元素取出并删除该元素
ls.remove(x)	将列表中出现的第一个元素x删除
ls.reverse(x)	列表ls中元素反转

列表类型

- 当使用一个列表改变另一个列表值时，Python不要求两个列表长度一样，但遵循“多增少减”的原则，例子如下。

```
In [43]: ls = list(range(6))
         ls
Out[43]: [0, 1, 2, 3, 4, 5]

In [44]: len(ls[3:])
Out[44]: 3

In [45]: 2 in ls
Out[45]: True

In [46]: ls[3] = 'replace'
         ls
Out[46]: [0, 1, 2, 'replace', 4, 5]
```

```
In [47]: ls[2:4] = ['machine','learning']
         ls
Out[47]: [0, 1, 'machine', 'learning', 4, 5]

In [51]: ls[2:4] = ['machine','learning','AI']
         ls
Out[51]: [0, 1, 'machine', 'learning', 'AI', 5]

In [52]: ls[2:4] = ['AI']
         ls
Out[52]: [0, 1, 'AI', 'AI', 5]
```

列表类型

- 与元组一样，列表可以通过for...in语句对其元素进行遍历，基本语法结构如下：

for <任意变量名> in <列表名>:
语句块

```
In [53]: for i in ls:
             print(i, end=" ")

0 1 AI AI 5
```

- 列表是一个十分灵活的数据结构，它具有处理任意长度、混合类型的能力，并提供了丰富的基础操作符和方法。当程序需要使用组合数据类型管理批量数据时，请尽量使用列表类型。

选择题 1分

设置

列表ls=[[125, 12],[[7,2],2,7],[5,2],1]，len(ls)值是多少?

A 2

B 4

C 8

D 9

提交

单选题 1分

设置

列表ls1=[125,1]，ls2=ls1, ls1[0] = 2, ls2值是多少?

A [125, 1]

B [2,1]

提交

示例：基本统计值计算

- 以最简单的统计问题为例，求解一组不定长数据的基本统计值，即平均值、标准差、中位数。
- 一组数据$S=s_0, s_1, \ldots, s_{n-1}$，其算术平均值、标准差分别表示为：
- $$m=\frac{\sum_{i=0}^{n-1} s_i}{n}, \qquad d=\sqrt{\frac{\sum_{i=0}^{n-1}(s_i-m)^2}{n-1}}$$
- 由于平均数、标准差和中位数是三个不同的计算目标，使用函数方式编写计算程序。
  - getNum()函数从用户输入获得数据
  - mean()函数计算平均值
  - dev()函数计算标准差
  - median()函数计算中位数

示例：基本统计值计算

- 以最简单的统计问题为例，求解一组不定长数据的基本统

In [54]:

```
from math import sqrt

def getNum(): #获取用户输入
    nums = []
    iNumStr = input("请输入数字(直接输入回车退出): ")
    while iNumStr != "":
        nums.append(eval(iNumStr))
        iNumStr = input("请输入数字(直接输入回车退出): ")
    return nums

def mean(numbers): #计算平均值
    s = 0.0
    for num in numbers:
        s = s + num
    return s / len(numbers)

def dev(numbers, mean): #计算方差
    sdev = 0.0
    for num in numbers:
        sdev = sdev + (num - mean)**2
    return sqrt(sdev / (len(numbers)-1))
```

```
def median(numbers): #计算中位数
    sorted(numbers)
    size = len(numbers)
    if size % 2 == 0:
        med = (numbers[size//2-1] + numbers[size//2])/2
    else:
        med = numbers[size//2]
    return med

n = getNum() #主体函数
m = mean(n)
print("平均值:{},方差:{:.2},中位数:{}.".format(m,\
 dev(n,m),median(n)))
```

```
请输入数字(直接输入回车退出): 12.2
请输入数字(直接输入回车退出): 11.8
请输入数字(直接输入回车退出): 10.9
请输入数字(直接输入回车退出): 11.7
请输入数字(直接输入回车退出): 13.1
请输入数字(直接输入回车退出): 12.7
请输入数字(直接输入回车退出):
平均值:12.066666666666665,方差:0.78,中位数:11.3.
```

列表类型

- 列表是一个动态长度的数据结构，可以根据需求增加或减少元素；
- 列表的一系列方法或操作符为计算提供了简单的元素运算手段；
- 列表提供了对每个元素的简单访问方式以及所有元素的遍历方式。

Python AI
组合数据类型

提纲

- 组合数据类型概述
- 列表类型与操作
- **jieba库的使用与词频统计**

Jieba库

- 英文单词间有空格：split()分词
- jieba是Python中一个重要的第三方中文分词函数库

import jieba

- jieba库是第三方库，不是安装包自带，需要通过pip指令安装

pip install jieba # pip3 install jieba

- 解析

函数	描述
jieba.cut(s)	精确模式，返回一个可迭代的数据类型
jieba.cut(s, cut_all=True)	全模式，输出文本s中所有可能单词
jieba.cut_for_search(s)	搜索引擎模式，适合搜索引擎建立索引的分词结果
jieba.lcut(s)	精确模式，返回一个列表类型，建议使用
jieba.lcut(s, cut_all=True)	全模式，返回一个列表类型，建议使用
jieba.lcut_for_search(s)	搜索引擎模式，返回一个列表类型，建议使用
jieba.add_word(w)	向分词词典中增加新词w

Jieba库

- 用法示例

```
In [73]: import jieba
         jieba.lcut("中华人民共和国是一个伟大的国家")
Out[73]: ['中华人民共和国', '是', '一个', '伟大', '的', '国家']

In [71]: jieba.lcut("中华人民共和国是一个伟大的国家", cut_all=True)
Out[71]: ['中华',
          '中华人民',
          '中华人民共和国',
          '华人',
          '人民',
          '人民共和国',
          '共和',
          '共和国',
          '国是',
          '一个',
          '伟大',
          '的',
          '国家']

In [72]: jieba.lcut_for_search("中华人民共和国是一个伟大的国家")
Out[72]: ['中华', '华人', '人民', '共和', '共和国', '中华人民共和国', '是', '一个', '伟大', '的', '国家']
```

文本文件打开和读取

- 文本文件打开、读取
- open() 函数用于打开一个文件，创建一个 file 对象

open(file, mode= 'r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True)

- 常用参数：
  - file : 一个包含了你要访问的文件名称的字符串值；
  - mode : mode 决定了打开文件的模式：只读，写入，追加等
  - Encoding：表示的是返回的数据采用何种编码，一般采用utf8或者gbk
- Read() 函数用于读取文件

词频统计：列表方法

- 仅使用列表进行词频统计

In [75]:

```
import jieba
txt = open("C:\RUC\课程\PythonAI\课程课件\高瓴人工智能学院简介.txt", "r", encoding='utf-8').read()
words = jieba.lcut(txt)
uniquewords = list()
counts = list()
for word in words:
    if len(word) == 1: #排除单个字符的分词结果
        continue
    else:
        if (word in uniquewords):
            tempindex = uniquewords.index(word)
            counts[tempindex] = counts[tempindex] + 1
        else:
            uniquewords.append(word)
            counts.append(1)

for i in range(len(counts)):
    print ("{0:<10}{1:>5}".format(uniquewords[i], counts[i]))
```

词频统计：列表方法

- 仅使用列表进行词频统计

```
过去        1
未来        3
已来        1
构建        1
人工智能        13
时代        2
宏大        1
世界观        1
影响        2
技术        2
发展        2
历史        1
趋势        1
吸纳        1
培养        2
领域        2
顶尖        1
学者        1
实践者        1
中国人民大学        4
```

词频统计-字典（dict）方法

- 词典方法词频统计：

In [74]:
```
import jieba
txt = open("C:\RUC\课程\PythonAI\课程课件\高瓴人工智能学院简介.txt", "r", encoding='utf-8').read()
words = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1: #排除单个字符的分词结果
        continue
    else:
        counts[word] = counts.get(word,0) + 1
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(15):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

```
人工智能        13
学院            8
一流            5
中国人民大学        4
未来            3
高瓴            3
院长            3
打造            3
```

```
人工智能        13
学院            8
一流            5
中国人民大学        4
未来            3
高瓴            3
院长            3
打造            3
全球            3
研究            3
联合            3
时代            2
影响            2
技术            2
发展            2
```

大作业1：

- 给定一个二维卷积核和由图片转换后的矩阵，实现二维卷积操作，观察卷积后的图片所产生的变化

- 提交：
  - 把Jupyter Notebook中的空缺部分填好，并运行得到结果
  - 提交文件

- 提交期限：2周

- 提交平台：未来课堂

关于卷积

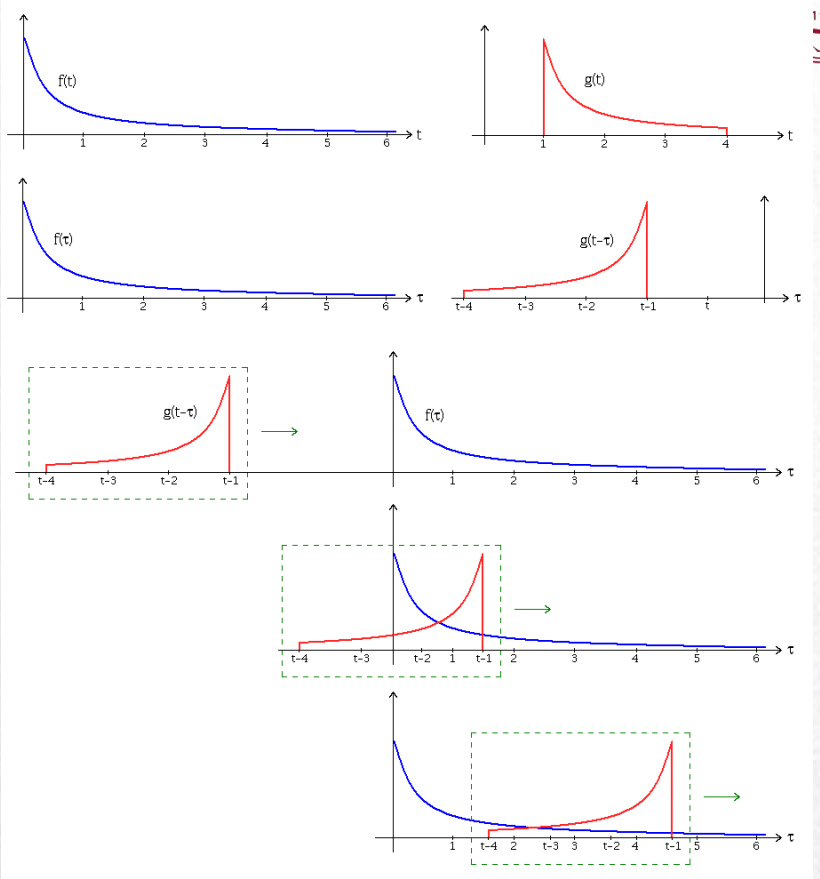
- 什么是卷积?
  - 两个函数f和g生成第三个函数的一种数学算子
  - f与经过翻转和平移的g的乘积函数所围成的曲边梯形的面积(滑动平均)

其连续的定义为:

$$(f*g)(n)=\int_{-\infty}^{\infty}f(\tau)g(n-\tau)d\tau$$

其离散的定义为:

$$(f*g)(n)=\sum_{\tau=-\infty}^{\infty}f(\tau)g(n-\tau)$$



离散卷积举例：2个骰子点数和为4的概率

- 两枚骰子
  - 骰子未必都是公平
    - 第一个骰子出现点i (i = 1, 2, ..., 6) 的概率为f(i)
    - 第二个骰子出现点i的概率为g(i)
  - 把这两枚骰子都抛出去
  - 求2个骰子**点数和为4**的概率
    - $(f * g)(4)$

f	1	2	3	4	5	6

f表示第一枚骰子
$f(1)$表示投出1的概率
$f(2)$、$f(3)$、$\cdots$以此类推

g	1	2	3	4	5	6

g表示第二枚骰子

应用卷积

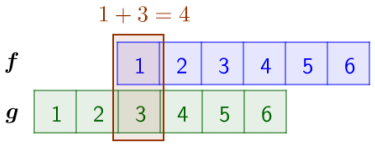
- 关键：两个骰子点数加起来等于
  - $(f * g)(4)$

$$f(1)g(3) + f(2)g(2) + f(3)g(1)$$

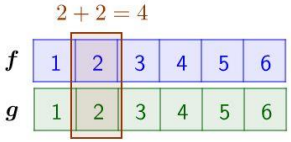
$$(f * g)(4) = \sum_{m=1}^{3} f(4-m)g(m)$$

  - $f * g(k)$：骰子点数和为k的概率

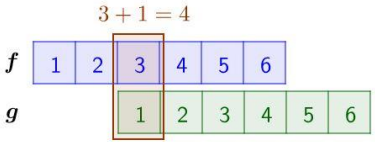
$$f * g(k) = \sum_{m=1}^{k-1} f(k-m)g(m)$$



出现概率为：$f(1)g(3)$



出现概率为：$f(2)g(2)$



出现概率为：$f(3)g(1)$

Python实现一维卷积

$$(f*g)(4)=\sum_{m=1}^{3} f(4-m)g(m)$$

- f和g分别为两个列表，存储点数对应的概率
  - 下标对应点数，注意下标从0开始
    - f[0]表示点数为1的概率：f[m - 1] = f(m)
  - 值为概率，暂时设置为等概率

- range(1, 4)：产生m =1 ， 2， 3

- conv += f[(4-m) - 1] * g[m - 1]
  - 乘加

```
f = [1/6, 1/6, 1/6, 1/6, 1/6, 1/6]
g = [1/6, 1/6, 1/6, 1/6, 1/6, 1/6]

conv = 0
for m in range(1, 4):
    conv += f[(4-m) - 1]*g[m - 1]
print(conv)
```

```
0.08333333333333333
```

谢谢！