人工智能程序设计

计算机学院

李晶晶

華南師範大學
South China Normal University

目录

- 数值数据类型
- 数据结构类型
  - 列表
  - 元组
  - 字典
  - 集合

数值数据类型

- 整数类型（int）
- 布尔类型（bool）
- 浮点类型（float）
- 复数类型（complex）

数据结构类型

- 根据某种方式将数据元素组合起来形成的一个数据元素集合
  - 序列
  - 映射
  - 集合

数据结构类型

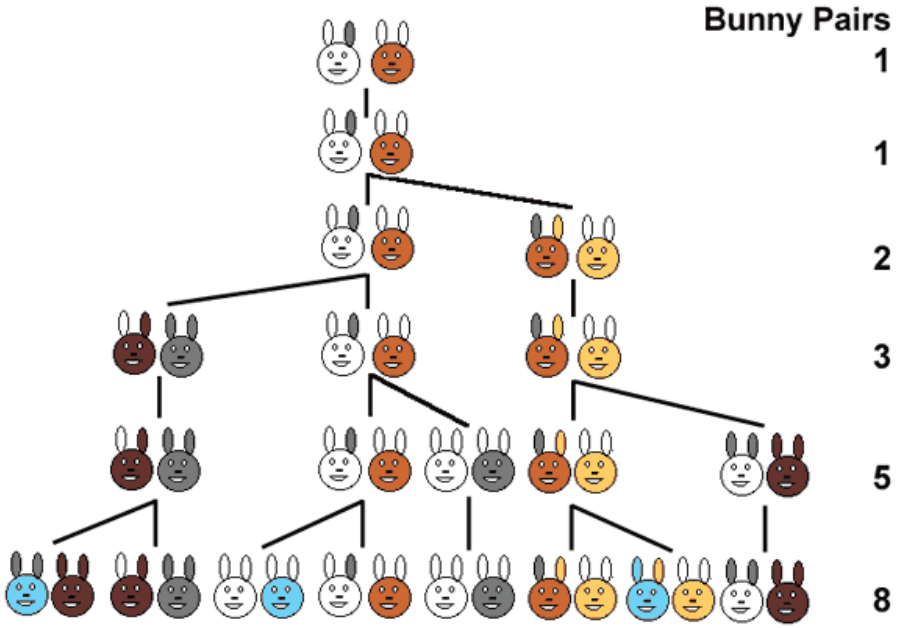
- 序列
  - 序列类型是一个元素向量，元素之间存在先后关系，通过序号访问，元素之间不排他。
- 映射
  - 映射类型是“键-值”数据项的组合，每个元素是一个键值对，表示为(key, value)。
- 集合
  - 集合类型是一个元素集合，元素之间无序，相同元素在集合中唯一存在。

数据结构类型

- 可变数据类型（列表、字典、可变集合）
  - 直接对数据结构对象的内容进行修改
  - 只对同一个对象进行操作
- 不可变数据类型（数字、字符串、元组、不可变集合）
  - 不能对数据结构对象的内容进行修改操作
  - 新旧对象两者引用两个不同的id地址值

问题1

- 问题描述
  - 已知前两项，计算出给定长度的斐波那契数列



问题1

- 问题分析
  - 输入：需要计算的数列项数
  - 处理：给定前两项：0和1

    运用f(n)=f(n-2)+f(n-1)(n>2, n为正整数）

    如果输入为浮点数或者负数或（n<=2)

    则提示错误
  - 输出：给定长度的斐波那契数列

列表（list）

- 列表是Python对象作为其元素并按顺序排列构成的有序集合。
- 创建列表
  - 方括号[ ]创建
  - 通过list()函数将元组、字符串或者集合转化成列表
  - 直接使用list()函数会返回一个空列表

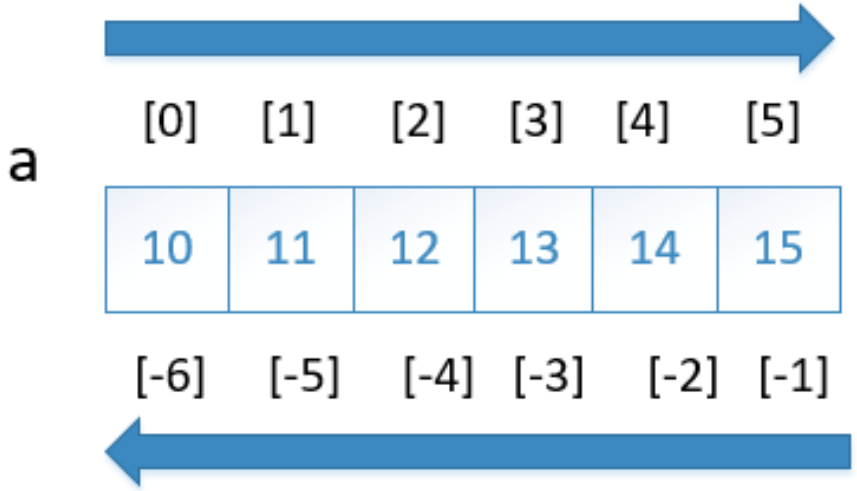
列表（list）

```
>>>ls = [1, 2.0, ['three', 'four', 5], 6.5, True]
#创建包含混合数据类型的嵌套列表
 >>>ls
[1, 2.0, ['three', 'four', 5], 6.5, True]
>>>empty_ls = []
>>>empty_ls
[]
>>>ls1 = list((425, "BIT", [10, "CS"], 425))
>>>ls1
[425, 'BIT', [10, 'CS'], 425]
>>>list("hello world")
['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd’]
>>>list()
[]
```

列表（list）

- 列表基本操作
  - ➢列表索引访问提取

```
>>>[1, 1.3, '2', 'China', ['I', 'am', 'another', 'list′ ]]
>>>L[0]
>>>L[4]
>>>L[4][2]
>>>len(L)
>>>L[len(L)-1]
>>>L[-4]
```



列表（list）

- 列表基本操作
  - ➢列表切片操作提取
    - ✓sequence_name[start:end:step]

```
>>>ls = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>>ls[2:7]
>>>ls[1:9:2]
>>>ls[-2:-8:-2]
>>>ls[1:4:0]
```

```
>>>ls[::]
>>>ls[:-7:-2]
>>>ls[6:]
>>>ls[1:8:-2]
>>>ls[::-1]
>>>ls[:]
```

列表（list）

- 列表基本操作
  - ➢列表切片操作提取
    - ✓sequence_name[start:end:step]

```
>>>ls = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>>ls[2:7]
[30, 40, 50, 60, 70]
>>>ls[1:9:2]
[20, 40, 60, 80]
>>>ls[-2:-8:-2]
[90, 70, 50]
>>>ls[1:4:0]
ValueError: slice step cannot be zero
```

```
>>>ls[::]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>>ls[:-7:-2]
[100, 80, 60]
>>>ls[6:]
[70, 80, 90, 100]
>>>ls[1:8:-2]
[]
>>>ls[::-1]
[100, 90, 80, 70, 60, 50, 40, 30, 20, 10]
>>>ls[:]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

列表（list）

- 列表基本操作
  - ➢增添列表元素
    - ✓append(x)在列表末尾添加一个元素

```
#<例3.1：列表append方法>
L = [1,1.3,"2","China",["I","am","another","list"]]
L.append(3)
print(L)  #输出
```

列表（list）

- 列表基本操作
  - ➢增添列表元素
    - ✓在列表末尾添加另一个列表

      extend(t)、+、+=

```
#<例3.2：列表拼接>
L = [1,1.3,"2","China",["I","am","another","list"]]
L1 = [4, 4, 4, "hello"]
L.extend(L1)
print(L)  #输出
L2 = [1, 2, 3, "这是L2"]
L2 += L1
print(L2)  #输出
```

列表（list）

- 列表基本操作
  - 增添列表元素

```
#<例3.3：列表insert方法>
L = [1,1.3,"2","China",["I","am","another","list"]]
L.insert(4, "insert" )
print(L)  #输出
L.insert(14, "insert" )
print(L)  #输出
```

  - **insert(i,x)在列表中添加一个元素，可以指定位置添加**
  - **插入位置超出列表尾端，则插入列表最后**

列表（list）

- 列表基本操作
  - 删除列表元素
    - ✓del语句 del list_name[i]
    - ✓pop语句 list_name.pop(i) 不指定位置，默认为-1
    - ✓remove语句 list_name.remove(x) x可以是元素位置或者指定元素

列表（list）

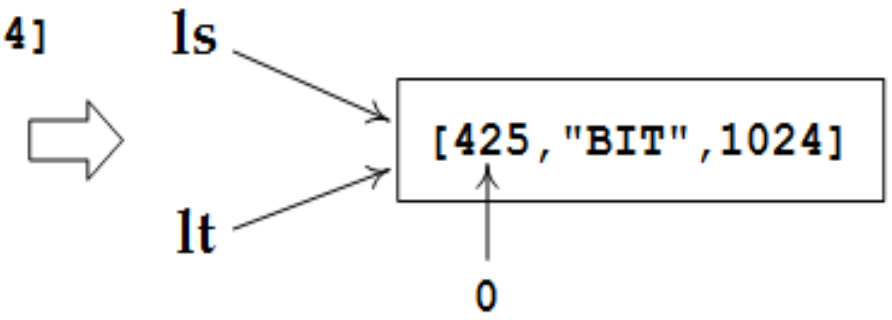
```
#<例3.4：列表删除方法>
L = [1,1.3,"2","China",["I","am","another","list"]]
del L[2]
print(L)  #输出
L1 = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]
del L1[0:2]
print(L1)
del L1[::2]
print(L1)
del L1[:]
print(L1)
del L1
```

```
L.pop(0)
print(L)  #输出
L.pop( )
print(L)
L.remove("China")
print(L)  #输出
```

列表（list）

- 列表基本操作
  - ➢修改列表元素
    - ✓列表必须通过显式的数据赋值才能生成，将一个列表赋值给另一个列表不会生成新的列表对象

```
>>>ls = [425, "BIT", 1024]
>>>lt = ls
>>>ls[0] = 0
>>>lt
```



列表（list）

- 列表基本操作
  - ➢修改列表元素
    - ✓通过切片来修改列表

```
>>>L1 = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]
>>> L1[0:3] = ["one", "two", "three"]
>>> L1
>>> L1[0:3] = []
>>> L1
>>> L1[::2] = ["#"]*6
>>> L1[:] = []
>>> L1
```

列表（list）

- 列表基本操作
  - ➢副本
    - ✓copy、切片操作、list( )

```
#<例3.5：列表副本>
a = [10, 20,30, 40, 50, 60, 70, 80]
b = a.copy(); c = a[:]; d = list(a); e = a
print(id(a), id(b), id(c), id(d), id(e))
a[0] = '修改了a'
b[0] = '修改了b'
print(a); print(b); print(c); print(d); print(e)
```

列表（list）

- 列表基本操作
  - ➢查询列表元素位置
    - ✓index来查询指定元素在列表中第一次出现的位置索引
    - ✓元素 in 列表对象元素至少在列表中出现过一次

```
a = [10, 20, 30, 40, 50, 60, 30, 80]
b = a.index(30)  #查询元素80在列表中第1次出现的位置
print(b)
c = 50 in a          #使用in函数判断列表是否包含元素
print(c)
```

列表（list）

- 列表其他常用操作
  - list.count
  - list.sort
  - sorted
  - list.reverse
  - len
  - *

列表（list）

- for 和range访问列表索引和值

```
list = [10,20,30,40,50,60,70,80,90]
for i in range(len(list)):
    print(f'{i}:{list[i]}')
```

- for e in L
  - e：可以是多变量

```
for x, y in [(1, 2), (2, 3), ("hello", "world")]:
    print(x, y)
for x, y, z in [(1, 2, 3), (2, 3, 4)]:
    print(x, y, z)
```

列表（list）

```
data = [num for num in range(20) if num%2==1]
```

- 列表解析式（列表推导式）
  - 高效创建新列表的方式，动态创建列表
  - Python迭代机制的一种应用
    - ✓使用列表推导式在一行代码中完成相同的任务
    - ✓映射：在列表推导式的表达式中执行操作
    - ✓过滤：在列表推导式中使用if子句
    - ✓用列表推导式处理另一个列表推导式

```
data = [i for i in range(6)]
```

```
data = list(range(6))
```

```
data = [2**i for i in range(64)]
```

```
num1 = ["one", "two", "three", "four", "five"]
num2 = [item.upper() for item in num1]
```

列表（list）

- 列表解析式（列表推导式）

```
[(i, j) for i in range(0,3) for j in range (0, 3)]
```

等价于

```
ls = []
for i in range(0, 3):
    for j in range(0, 3):
        ls.append((i, j))
```

```
[(i, j) for i in range(0, 3) if i<1 for j in range(0, 3) if j>1]
```

等价于

```
ls = []
for i in range(0, 3):
    if i < 1:
        for j in range(0, 3):
            if j > 1:
                ls.append((i,j))
```

列表（list）

- 添加列表元素

  问题：遍历列表L，并打印出L中所有元素，还要在元素为0时向列表中添加元素100，使用“for e in L：”结构该如何实现呢？

  - 方法一：直接使用append函数在原列表中添加新元素100

```
#<例3.6：列表中添加元素100（append）>
L=[0,1,2,3,4,5]
for e in L:
    print(e,end=' ')
    if e==0: L.append(100)
```

列表（list）

- 添加列表元素

  问题：遍历列表L，并打印出L中所有元素，还要在元素为0时向列表中添加元素100，使用“for e in L：”结构该如何实现呢？

  ➢ 方法二：使用L=L+[100]这种方式添加新元素100

```
#<例3.7：列表中添加元素100（+）>
L=[0,1,2,3,4,5]
for e in L:
    print(e, end=' ')
    if e == 0: L = L + [100]
```

列表（list）

- 添加列表元素

  问题：遍历列表L，并打印出L中所有元素，还要在元素为0时向列表中添加元素100，使用“for e in L：”结构该如何实现呢？

- 方法三：使用列表的专有方法insert()在原列表中插入新元素100

```
#<例3.8：列表中添加元素100（insert）错误>
L=[0,1,2,3,4,5]
for e in L:
    print(e,end=' ')
    if e==0:L.insert(0,100)
```

列表（list）

- 添加列表元素

  问题：遍历列表L，并打印出L中所有元素，还要在元素为0时向列表中添加元素100，使用“for e in L：”结构该如何实现呢？

- 方法三：使用列表的专有方法insert()在原列表中插入新元素100

```
#<例3.9：列表中添加元素100（insert）正确>
L=[0,1,2,3,4,5]
for e in L:
    print(e,end=' ')
    if e==0:L.insert(len(L),100)
```

列表（list）

- 综上所述，原列表在循环中被改变是危险的，难以预料的。
- 所以为了避免发生不可预料的情况，尽量不对原列表L进行改变，为了安全起见，还是新设一个列表。

列表（list）

- 添加列表元素
  - 开销

```
# <例3.10：append()和“L+[i]”在时间开销方面的对比>
import time
def test_time(k):
    print("***** k=", k)
    L = []; start = time.perf_counter()
    for i in range(k):
        L.append(i)
    elapsed = time.perf_counter() - start
    print("使用append花时间：", elapsed)
    start = time.perf_counter()
    for i in range(k):
        L = L + [i]
    elapsed = time.perf_counter() - start
    print("使用L=L+[i]花时间：", elapsed)
test_time(5000)
test_time(10000)
test_time(20000)
test_time(40000)
```

列表（list）

- 删除列表元素

```
#<例3.11：移除列表中为0的元素_2（错误）>
L=[0,0,1,2,3,0,1]
i=0
while i < len(L):
    if L[i]==0: L.remove(0)
    i+=1
print(L)
```

列表（list）

- 删除列表元素
  - 方法一：

```
#<例3.12：移除列表中为0的元素改进1>
L=[0,0,1,2,3,0,1]
i=0
while 0 in L:
    L.remove(0)
print(L)
```

列表（list）

- 删除列表元素
  - 方法二：

```
#<例3.13：移除列表中为0的元素改进2>
L=[0,0,1,2,3,0,1]
i=0
while i< len(L):
    if L[i]==0: L.remove(0)
    else: i+=1
print(L)
```

列表（list）

- 删除列表元素
  - 方法三：

```
#<例3.14：移除列表中为0的元素改进3>
L=[0,0,1,2,3,0,1]
L1=[]
for e in L:
    if e!=0: L1.append(e)
print(L1)
```

列表（list）

- 删除列表元素
  - ➢时间开销
    - ✓测试方法一

```
#<例3.15：将列表中的0去掉方法1>
import time  #引入time模块
start=time.clock()
L=[1 for i in range(1000000)]+[0 for i in range(1000)]
while 0 in L:
    L.remove(0)
elapsed=time.clock()-start
print("方法一： ")
print("用while 0 in L, remove后列表长度=",len(L),"花时间： ", elapsed)
```

列表（list）

- 删除列表元素
  - ➢时间开销
    - ✓测试方法二

```
#<例3.16：将列表中的0去掉方法2>
import time  #引入time模块
start=time.clock()
L=[1 for i in range(1000000)]+[0 for i in range(1000)]
i=0
while i < len(L):
    if L[i]==0: L.remove(0)
    else: i+=1
elapsed=time.clock()-start
print('方法二')
print("一个个遍历再remove后，列表长度=",len(L),"花时间： ", elapsed)
```

列表（list）

- 删除列表元素
  - ➢时间开销
    - ✓测试方法三

```
#<例3.17：将列表中的0去掉方法3>
import time  #引入time模块
start=time.clock() #程序开始运行时的时间
L=[1 for i in range(1000000)]+[0 for i in range(1000)]
L1=[]
for e in L:
    if e!=0: L1.append(e)
elapsed=time.clock()-start #循环结束消耗的时间
print('方法三')
print("新列表L1的长度：",len(L1), "花时间：", elapsed)
```

列表（list）

- 生成列表的技巧
  - ➢生成矩阵

矩阵：一维矩阵（如[1,2,3]、[aa,bb,cc] ）

二维矩阵（如[[1,2,3],[4,5,6]] ）

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

✓生成一维矩阵：生成一个一维矩阵，长度为100，每个元素都为0

方法一：[0]*100

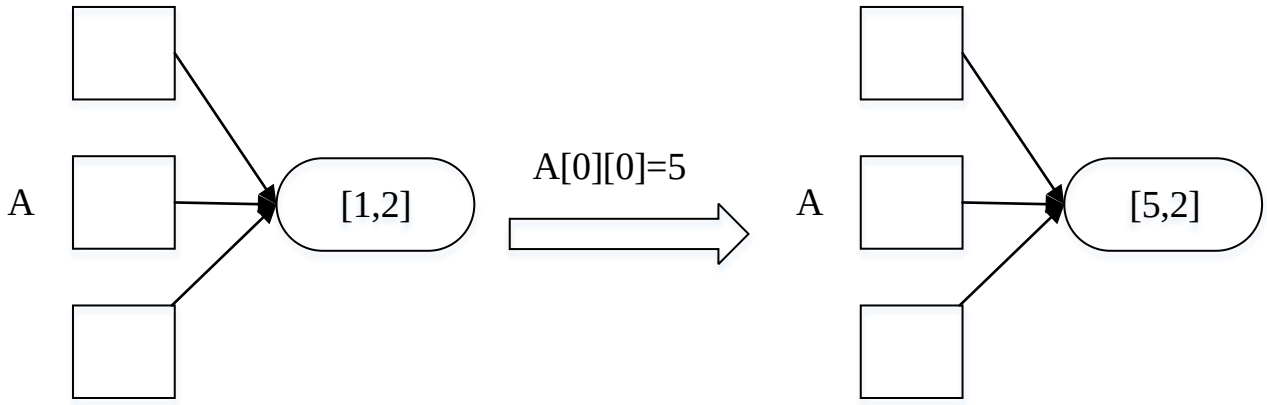
方法二：[0 for i in range(100)]

列表（list）

- 生成列表的技巧
  - ➢生成矩阵
    - ✓二维矩阵：生成一个二维矩阵， A=[[1,2],[1,2],[1,2]]

    方法一： A=[[1,2]]*3

    注意，这种方式有一个致命的问题，当执行A[0][0]=5的时候，矩阵会变成 A=[[5,2],[5,2],[5,2]]

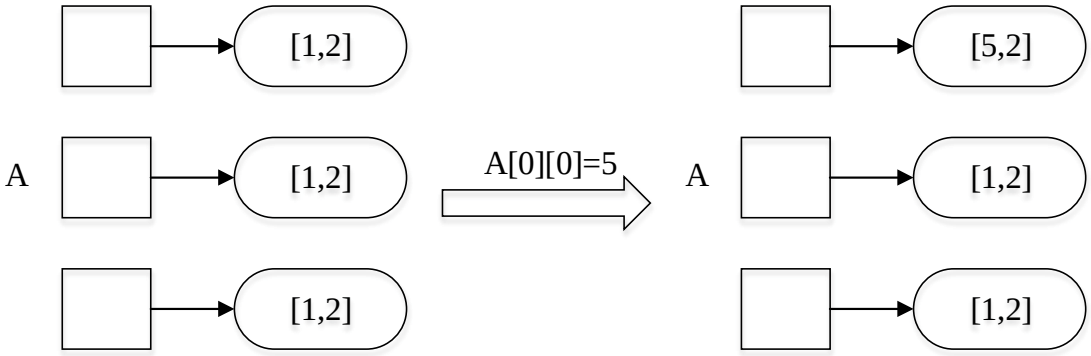


列表（list）

- 生成列表的技巧
  - ➢生成矩阵
    - ✓二维矩阵：生成一个二维矩阵， A=[[1,2],[1,2],[1,2]]

    方法二： A=[[1,2] for i in range(3)]

    推荐使用这种列表解析表达式的方式生成二维矩阵



问题1

- 演示代码
  - 输入：需要计算的数列项数
  - 处理：给定前两项：0和1，运用f(n)=f(n-2)+f(n-1)(n>2, n为正整数）

    如果输入为浮点数或者负数或者n<=2则提示错误
  - 输出：给定长度的斐波那契数列

上机实践

- 求等比数列的前n项之和。例如：一个首项为2，等比为2， 项数为5的数列2，4，8，16，32， 它的和为2+4+8+16+32=62.
  - 输入：首项a0，等比p，项数n（都为正整数）
  - 处理：第i项为a0*p**i

    各项之和几位累加的前i项之和

    如果输入为浮点数或负数则提示错误
  - 输出：等比数列以及等比数列之和