Lesson 3作业

2020年11月14日

23:16

## （一）列表

1. **Python**
   1. 实例化
      1. list=[1,2,3,4,5] ##列表名=[逗号分隔不同数据项]
      2. list1 = [1, 3, 5, 8, 11, 'j']
      3. 访问列表中的元素
         1. print（list[元素索引]）
         2. 元素索引从0开始，也可以倒数，最后一个元素是-1
      4. 使用列表中的值
         1. "He 110" + 
            helloword 
            print(helloword) 
            HelloQwe. 
            ase 
            namel ist[0]. title +" 
   2. 增加元素
      1. list=[] 列表名=[列表内容]
      2. list.append (obj) 列表名.append（添加元素）——在列表末尾添加新元素
         1. 计算机生成了可选文字:
            >>listl.append〔'j')
            >>PI、int(listl)
      3. list.extend(seq) 列表名.extend（添加另一已定义好的列表）——在列表末尾添加序列（多个值）e.g list1.extend(list2)
         1. 计算机生成了可选文字:
            [1，
            1i5t1：[1，3，5]
            1i5t2：[8，11]
            1i5t1.extend（1i5t2）
            (list1)
            print
            3，5，8，11]
      4. list.insert(index,obj) 在列表中index位置插入元素 e.g list1.insert(1,'v')
         1. listl. insert (1, ' w' ) 
            pr9-1,t (l F,tl) 

1. 对列表进行检索
   1. list[a] a为列表中元素索引，找到列表中位置为a的元素
   2. list[a:b] a,b为列表中元素的索引,找到列表中从a到b之前的元素
      1. print (listl 
         print (Iist2) 
         list2-l 21 
         prlnt (Ilst2) 
         print [O: 2]) 
   3. list.count(obj) 列表名.count（元素），统计某个元素在列表中出现的次数
      1. list I 
         llstl. eoLt-1t 
         'b' 
   4. list.index(obj) 列表名.index（元素），找出某个元素在列表中的第一个匹配项的位置
      1. 
2. 删除列表中的元素
   1. del list[] 用del删除列表中某个位置的值,中间空格连接
      1. del Ilstl 
         print (listl) 
   2. list.pop(index) 移除列表中位置为index的元素，并返回该元素的值，默认index=-1，即移除列表中最后一个元素，并返回最后一个元素的值
      1. Ilstl. pop (1) 
         (Ilstl) 
         pop C) 
         print (l istl) 
   3. list.remove(obj) 移除列表中某个obj的第一个匹配项
      1. llstl-[ 
         listl. reınove b' 
         prŞn,t (l isti) 
3. 修改列表
   1. 修改列表中的元素
      1. list[]="更改后的元素" ##list[要修改的元素的索引] = ‘修改后的元素名’
      2. print (helloword) 
         Hel loawe. 
         prinț (namelist) 
         uio' , 
         rty' , 
         asc' ) 
   2. list.reverse() 反向列表中的元素
      1. 1 Istl. rewerse C) 
         (l țsțl) 
   3. list.sort() 默认对列表进行升序排序，永久排序
      1. 参数: reverse reverse=True则对列表进行降序排序，默认升序；key 指定排序标准的函数
      2. list]. sort 
         listl. sort (reverse-True) 
         (list l) 
   4. list.sorted() 对列表进行临时排序 默认升序，True为降序
      1. (narnelist) 
         uio' , 
         rty', ' 
         asc' I 
         prit# (sort$d (namelist)) 
         asc' , 
         rty' , 
         uio' J 
         print (sorted (namelist, reverse- 
         -True)) 
         t' ui0', ' rty 
         asc' J 
   5. len（列表名） 确定列表长度
      1. 计算机生成了可选文字:
         >>>(namelist)
         3
4. **R**
   1. 列表实例化
      1. mylist <- list (object1,object2,…)
      2. 可以对列表中的对象命名
         1. mylist <- list(name1=object1 , name2=object2 , …)
      3. R中列表下标从1开始
   2. 访问列表中元素
      1. 使用列表下标访问列表中的第n个元素 print（列表名[下标]）
         1. > prfnt(n1fstL52) 
      2. 使用元素名访问元素（如果在实例化过程中命名了）
         1. 列表名$元素名
            1. "Google" "Runoob" "Taobao" 
   3. 增加元素
      1. 将值赋给数据结构中不存在的元素时，R会自动扩展数据结构以容纳新值，并用NA填充空位
      2. •my list[aJ 
         print(nyli5t) 
   4. 修改列表，更新元素
      1. fix（object） 直接编辑对象
      2. newobject <- edit (object) 编辑对象并另存为newobject（新对象名）
      3. list[序号] <- 新元素
         1. assLAu)wL.ud 
            :e6essau Su L 
   5. 删除元素
      1. list[要删除的元素索引序号] <- NULL
      2. •my list[aJ 
         print(nyli5t) 
   6. 合并列表 用c（list1，list2）
      1. 
5. **Julia** 
   1. 实例化
      1. [item1, item2, ...]

mylist 
Eelement Array {Any, 1} 
"b" 
2 

1. 检索
   1. 通过索引检索数据片段（Julia的索引也是从1开始，此三种中只有python从0开始）
      1. list[]
         1. julla> mylist 3 
2. 修改
   1. 用索引编辑列表中的元素
      1. list[需修改元素索引] = 修改后元素内容
         1. ulia> 
            "f" 
            ulia> 
            -Lia 
            mylist C 3) 
            print (m;list) 
3. 增加
   1. 函数 push！ 在队列末尾加入元素，
      1. julia> push! (mylist, 
         5—e1ement Array (Any, If 
         2 
4. 删除
   1. 函数pop！删除队列末尾的元素，同时输出该元素
      1. Jllia) pop! (mylist) 
         tillia) print (mylist) 
5. 多维列表
   1. 如例，简单地在 []中，套[]区分
      1. julia> print (mylist) 
         yt"a", " " , 2) 
         julia> 2, 3), (4, 5), (2, 3, 4, 5]] 
         -element Array {Array {Int64, 1}, 1} 
         [1, 2, 31 
         [4, 51 
   2. 函数Rand（行数，列数） 用随机值填充列表
      1. 3X5 Array{F10at64, 2} 
         o. 553206 
         o. 0633517 
         o. 401473 
         o. 998094 
         o. 306895 
         o. 671703 
         o. 752675 
         o. 407543 
         o. 0224736 
         o. 757695 
         o. 671774 
         o. 3857 
         o. 439177 
         o. 551673 
         o. 0167395 
   3. 复制列表
      1. 复制要使用 新列表名=copy（旧列表名）的形式，而不可直接使用新列表=旧列表。后者并未复制一个独立的列表，编辑其中之一会导致另一也发生改变
      2. 'Lilia) print (ml list) 
         ILIlia> nlist=mylist 
         -element Array (Any, 1} : 
         2 
         ILIlia> pop! (nlist) 
         ILIlia> print (m! list) 
         ILIlia> 

jlllia> print 
jlllia> 
—element Array (Any, l} : 
julia> push! (nlist, "l " ) 
—element Array (Any, l} : 
jillia> 
jlllia> 
nyc a 
illlia) 
prln! i st) 
print (nli st) 

## （二）元组实例化

1. Julia

* 1. （item1，item2，…）
  2. 元组可以通过下标查找，但是不能更新
  3. Ilia> l, 2) 
     Tl [3] 

1. python
   1. 元组使用（），列表用[]
      1. tup2 2, 3, 4, ) 
         print (tup2[O]) 
   2. 元组不可修改，但可连接组合，使用 新元组=元组1+元组2
2. R
   * 1. 元组在R中相当于固定的c()？？