Python基础课 列表+字典

一、列表

列表是按照特定顺序的排列组合，就像数学中的数列，列表中的元素具有一定的排列顺序。

在Python中，列表用方括号[ ]来表示列表，比如：

```
>>>a= ['Python','C','Java']
```

1、访问列表中的元素

索引开始：0

如果我们想要打印上述列表中的Python，就需要我们访问列表中的第一个元素。在Python中，列表的访问从0开始，索引数为元素的位置减去1，访问的元素位置放在方括号里面，如果我们想要访问第一个元素Python，则索引为0，即：

```
>>>a=['Python','C','Java']
>>>print(a[0])
Python
```

或者，在python中也可以逆序访问，从后往前索引依次是-1 -2 ...，比如

```
>>>print(a[-3])
```

```
Python
```

修改列表元素

在刚刚的列表中，如果我想把列表a中的第三个元素Java修改为R，则可以采用如下的命令：

```
>>>a[2]='R'  #Java在原来的列表中位置为3，所以索引为3-1=2
```

直接用赋值的命令(=)来修改列表中的元素，修改后，再试着打印原来的列表，看有没有修改成功。

```
>>>print(a)
['Python', 'C', 'R']
```

在列表中添加元素：append( ) insert( )

在列表末尾添加元素

在刚刚的列表末尾添加元素Ruby，使用方法append( )

```
>>>print(a)
['Python', 'C', 'R']

>>>a.append('Ruby')
>>>print(a)
['Python', 'C', 'R', 'Ruby']
```

在列表中添加元素

如果想把元素Ruby添加到Python后面，则需要使用**方法insert()。**

```
>>>a=['Python', 'C', 'R']  #  重新定义列表，避免上一步的操作
>>>a.insert(1,'Ruby')  # 将Ruby添加到Python后
>>>print(a)
['Python', 'Ruby', 'C', 'R']
```

从列表中删除元素：del语句，remove()，pop()

知道元素的位置，删除某个特定位置的元素用**del语句**

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>print(a)
['Python', 'Ruby', 'C', 'Java']

>>>del a[1] # 删除刚刚添加的Ruby
>>>print(a)
['Python', 'C', 'Java']
```

不知道元素的位置，但是知道要删除什么元素，使用**方法remove()**

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>print(a)
['Python', 'Ruby', 'C', 'Java']

>>>a.remove('Ruby')
>>>print(a)
['Python', 'C', 'Java']
```

如果想删除这个值并且继续使用它，可以使用**方法pop()**，方法pop( )可以将原有列表中特定元素删除，并且可以将之赋值给新的变量。

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>print(a)
['Python', 'Ruby', 'C', 'Java']

>>>b=a.pop(1)
>>>print(a)
['Python', 'C', 'Java']

>>>print(b)
Ruby
```

2、组织列表

组织列表：方法sort()，函数sorted()， 方法reverse()

列表的永久性排序sort()

使用**方法sort()**可以对列表进行永久性排序

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>print(a)
['Python', 'Ruby', 'C', 'Java']

>>>a.sort()  # 按照字母顺序排序
>>>print(a)
['C', 'Java', 'Python', 'Ruby']

>>>a.sort(reverse=True)  #按照字母逆序排序
>>>print(a)
['Ruby', 'Python', 'Java', 'C']
```

列表的临时排序sorted()

使用**函数sorted()**可以对列表进行临时性排序

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>print(a)
['Python', 'Ruby', 'C', 'Java']

>>>sorted(a)  # 临时性排序
>>>print(a)  #看原来的列表排序有无改变
['Python', 'Ruby', 'C', 'Java']

>>>print(sorted(a))  # 打印出临时性排序的列表
['C', 'Java', 'Python', 'Ruby']
```

倒着打印列表

要倒着打印列表，可以用**方法reverse()**，方法reverse永久性的修改了排列的元素

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>print(a)
['Python', 'Ruby', 'C', 'Java']

>>>a.reverse()
>>>print(a)
['Java', 'C', 'Ruby', 'Python']
```

确定列表的长度

可以用**函数len()**确定列表的长度

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>len(a)
4
```

使用列表时避免索引错误

常见的索引错误包括：

1、忘记索引数是元素位置减去1

2、超出访问的列表索引

3、列表为空列表时，倒着访问列表a[-1]错误

注：当发生索引错误时，可以将列表长度打印出来观察是什么错误

3、操作列表——创建数值列表

使用**range( )**创建数字列表

可以使用**函数range( )** 和**函数list ( )** 创建数字列表；

函数range( ) 可以生成一系列的数字，里面第一个参数是起始值，第二个参数表示*不超过这个值的终止值*，第三个参数表示步长，默认为1；

函数list( ) 可以将里面的参数转换为列表。

```
>>>c=list(range(1,10))
>>>print(c)
[1, 2, 3, 4, 5, 6, 7, 8, 9]

>>>d=list(range(1,10,2))
>>>print(d)
[1, 3, 5, 7, 9]
```

对数字列表进行简单的统计计算

可以对数值列表进行简单的统计，例如最大值、最小值、总和。

```
>>>c=list(range(1,10))
>>>min(c)
Out[34]: 1

>>>max(c)
Out[35]: 9

>>>sum(c)
Out[36]: 45
```

4、操作列表——使用列表的一部分

切片

切片，相当于把列表其中的一部分切出来。要创建切片，可以指定第一个元素和最后一个元素的位置，切片到达第二个元素的前一个元素停止切片，类似于函数range( )

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>print(a[1:3])  #切片第2到第4
['Ruby', 'C']

>>>print(a[:2])  #不指定第一个元素，从开始切片到第3个元素
['Python', 'Ruby']

>>>print(a[2:])  #不指定最后一个元素，从第3切片到最后
['C', 'Java']

>>>print(a[-3:])  #倒着切片，倒数第三个到最后一个
['Ruby', 'C', 'Java']
```

复制列表

使用切片的方法来复制列表，会产生两个列表。如果使用赋值（=）的方法来复制列表，第二个列表仅仅是指向第一个列表，并没有复制。

```
>>>a=['Python', 'Ruby', 'C', 'Java']
>>>b=a[:]
>>>a.append('CSS')
>>>print(a)
['Python', 'Ruby', 'C', 'Java', 'CSS']

>>>print(b)
['Python', 'Ruby', 'C', 'Java']
```

采用赋值的方法来复制列表

```
>>>a=['Python', 'Ruby','C', 'Java']
>>>b=a
>>>a.append('CSS')
>>>print(a)
['Python', 'Ruby', 'C', 'Java', 'CSS']

>>>print(b)
['Python', 'Ruby', 'C', 'Java', 'CSS']
```

二、字典

字典是 Python 内置的一种数据结构，它便于语义化表达一些结构数据，字典是开发中常用的一种数据结构

1、字典介绍

1. 字典使用花括号 {} 或 dict 来创建，字典是可以嵌套使用的
2. 字典是成对出现的，字典以键 (key) 值 (value) 对形式体现

3. 键与值之间用冒号 : 分隔，每个键值对之间用逗号，分隔开
4. 字典的 key 是唯一的，而 value 可以重复出现
5. 字典的 key 不使用中文或其他字符，这是业内约定俗成的做法

2、创建字典

使用花括号 {} 创建字典

注意：字典每个键值对之间要用逗号，分隔开

```
emp = {'name':'张三' ,  'age':22 , 'sex':'男'}
  print(emp)
# 运行结果：{'name':'张三' ,  'age':22 , 'sex':'男'}
# 字典打印时会将花括号也打印出来
  print(type(emp))
# 运行结果：<class 'dict'>
# 从以上打印类型可以看出变量属于 dict 字典类型/3、
```

3、字典取值

> 字典取值有两种方式

1. 方式一

在字典变量后面使用方括号传入字典的 key 进行取值

> 这种方法有一个弊端：如果字典中不存在 key， 会报 KeyError 错误

```
emp = {'name':'张三' , 'age':22 , 'sex':'男'}
print(emp['name'])
# 运行结果：张三
```

2. 方式二

> 使用字典 get 方法取值

如果不存在 key，返回 None 或自己指定的值，例如 N/A，以下为示例代码

```
emp = {'name':'张三' , 'age':22 , 'sex':'男'}
v = emp.get('name')
print(v)
# 运行结果：张三
```

```
emp = {'name':'张三' , 'age':22 , 'sex':'男'}
v = emp.get('dept','其他部门')
print(v)
# 运行结果：其他部门
```

4、字典的操作

新增

列表的新增操作和更新操作基本相同，Python 字典秉承：“有则更新，无则新增”原则

当字典存在对应的 key 时执行更新，当字典不存在对应的 key 时执行新增

1. 新增操作

```
emp = {'name': '张三', 'age': 22, 'sex': '男', 'dept': '研发部'}
emp['job'] = '销售'
print(emp)
# 运行结果：{'name': '张三', 'age': 22, 'sex': '男', 'dept': '研发部',
'job': '销售'}
```

2. 批量操作

```
emp = {'name': '张三', 'age': 22, 'sex': '男', 'dept': '研发部'}
emp.update(dept='推广部' , job='推广员')
print(emp)
# 运行结果：{'name': '张三', 'age': 22, 'sex': '男', 'dept': '推广部',
'job': '推广员'}
```

删除

1. 使用 pop() 方法删除

> 该方法删除字典给定的键及对应的值
> pop() 可返回值，返回值为被删除的值

```python
emp = {'name': '张三', 'age': 22, 'sex': '男', 'dept': '研发部'}
dept = emp.pop('dept')

print(emp)
# 运行结果：{'name': '张三', 'age': 22, 'sex': '男'}

print(dept)
# 运行结果：研发部
```

2. 使用 popitem() 删除字典最后一个 kv

popitem() 返回的是一个元组

```python
emp = {'name': '张三', 'age': 22, 'sex': '男', 'dept': '研发部'}
kv = emp.popitem()

print(emp)
# {'name': '张三', 'age': 22, 'sex': '男'}

print(kv)
# 运行结果：('dept', '研发部')
```

3. 清空字典 clear()

```python
emp = {'name': '张三', 'age': 22, 'sex': '男', 'dept': '研发部'}
emp.clear()
print(emp)
# 运行结果：{}
```