Python 程序设计基础

Python Programming

初识列表

- Python 提供了一种被称为列表的数据类型，可以存储任意大小的数据集合。
- 列表是序列类型。列表是任何元素的序列。列表既可以包含同类型的元素也可以包含不同类型的元素。
- 列表中的元素用逗号分隔并且由一对中括号 [ ] 括住。
  - list1 = []　# 空列表
  - list2 = [2, 3, 4]
  - list3 = ["red", "green", "blue"]
  - list4 = [2, "three", 4.0]

初识列表

- 列表中的元素通过“列表名 [下标]” 来访问。列表下标从 0 开始。
  - list1[0] 是列表 list1 的第一个元素，而 list1[9] 是列表 list1 的最后一个元素。
  - 下标越界访问列表会导致“ IndexError” 异常。 <class 'list'> 表示列表类型。

```
>>> list1 = [5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 45.45, 99.993, 11123]
>>> type(list1)
<class 'list'>
>>> list1[0]
5.6
>>> list1[9]
11123
>>> list1[10]
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    list1[10]
IndexError: list index out of range
>>> list1[0] = 6.5
>>> list1
[6.5, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 45.45, 99.993, 11123]
>>>
```

初识列表

- 使用内置函数
  - len 函数返回列表的元素个数
  - max 函数和 min 函数分别返回列表（元素必须是相同类型）中的最大值元素和最小值元素
  - sum 函数返回列表（元素为数字）中所有元素的和

```
>>> list1 = [2, 3, 4, 1, 32]
>>> len(list1)
5
>>> max(list1)
32
>>> min(list1)
1
>>> sum(list1)
42
>>>
```

初识列表

- split 方法
  - 字符串中有 split 方法，将字符串（默认以空格分隔）分解成其子串组成的列表。

```
>>> items = "Jane John Peter Susan".split()
>>> items
['Jane', 'John', 'Peter', 'Susan']

>>> items = "1 2 3".split()
>>> items
['1', '2', '3']
>>>

>>> items = "2016/11/11".split('/')
>>> items
['2016', '11', '11']
```

计算平均值的程序

要求用户在一行上输入三个整数，其间以空格间隔。计算并输出它们的平均值。

```
# 计算平均值
# 输入三个整数
# 将以空格分隔的三个整数构成的字符串分解列表
line = input("请输入以空格分隔的三个整数: ").split()
# 将列表中的元素转换为数值
number1 = eval(line[0])
number2 = eval(line[1])
number3 = eval(line[2])
# 求平均值，计算结果存放在变量average中
average = (number1 + number2 + number3) / 3
# 输出平均值
print(number1, number2, number3, "的平均值是", average)

请输入以空格分隔的三个整数: 1 2 3
1 2 3 的平均值是 2.0
>>>
```

计算平均值的程序

要求用户在一行上输入三个整数，其间以空格间隔。计算并输出它们的平均值。

使用 map 函数可以将序列映射并解包，可以用于输入数据，比前面例子简单。

```
# 计算平均值
number1, number2, number3 = map(int, input().split())
average = (number1 + number2 + number3) / 3
print(number1, number2, number3, "的平均值是", average)
```

```
1 2 3
1 2 3 的平均值是 2.0
```

初识元组

- 与列表类似，元组也是序列类型。元组是任何元素的序列。元组既可以包含同类型的元素也可以包含不同类型的元素。
- 元组中的元素用逗号分隔并且由一对圆括号 ( ) 括住。
  - tuple1 = () # 空元组
  - tuple2 = (2, 3, 4)
  - tuple3 = ("red", "green", "blue“)
  - tuple4 = (2, "three", 4.0)
- 创建只有一个元素的元组时，要在元素后面加上一个逗号，否则创建的是整数对象，不是元组。

```
>>> tuple5 = (1,)
>>> type(tuple5)
<class 'tuple'>
```

```
>>> tuple6 = (1)
>>> type(tuple6)
<class 'int'>
```

- 元组和列表的主要区别：元组被创建后，就无法直接修改元组中的元素值。

初识元组

- **元组中元素通过：元组名 [下标] 来访问。元组下标从 0 开始。**
  - **tuple1[0] 是元组 tuple1 的第一个元素，而 tuple1[9] 是元组 tuple1 的最后一个元素**
  -

```
>>> tuple1 = (5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 45.45, 99.993, 11123)
>>> type(tuple1)
<class 'tuple'>
>>> tuple1[0]
5.6
>>> tuple1[9]
11123
>>> tuple1[10]
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    tuple1[10]
IndexError: tuple index out of range
>>> tuple1[0] = 6.5
Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    tuple1[0] = 6.5
TypeError: 'tuple' object does not support item assignment
>>>
```

le'> 表示元组类型。

初识元组

- 使用内置函数
  - len 函数返回元组的元素个数
  - max 函数和 min 函数分别返回元组（元素必须是相同类型）中的最大值元素和最小值元素
  - sum 函数返回元组（元素为数字）中所有元素的和

```
>>> tuple1 = ("red", "green", "blue")
>>> len(tuple1)
3
>>> max(tuple1)
'red'
>>> min(tuple1)
'blue'
>>> tuple2 = (7, 1, 2, 23, 4, 5)
>>> sum(tuple2)
42
>>>
```

初识字典

- 字典是一系列键 / 值对的集合，每个键都与一个值相关联，可以使用键来访问与之相关联的值。
- 键 / 值对使用冒号分隔。各键 / 值对之间用逗号分隔并由一对花括号 {} 括住。
  - dict1 = {}　# 空字典
  - dict2 = {"one":1, "two":2, "three":3}
- len 内置函数获取字典的长度。
- 字典名 .get(key) ，返回键 key 对应的值，若键不存在，则返回 None 。
- 字典名 [key] ，访问键 key 对应的值，若键不存在，会导致“ KeyError” 异常。
- 字典名 [key]=value ，修改键 key 对应的值，若键不存在，则将 key:value 添加到字典中。

初识字典

```
>>> dict1 = {"one":1, "two":2, "three":3}
>>> type(dict1)
<class 'dict'>
>>> dict1.get("one")
1
>>> dict1.get("four")
>>> dict1["four"] = 4
>>> dict1
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
>>> dict1["one"] = '1'
>>> dict1
{'one': '1', 'two': 2, 'three': 3, 'four': 4}
>>> dict1["five"]
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    dict1["five"]
KeyError: 'five'
>>>
```

- **<class 'dict'> 表示字典类型。**