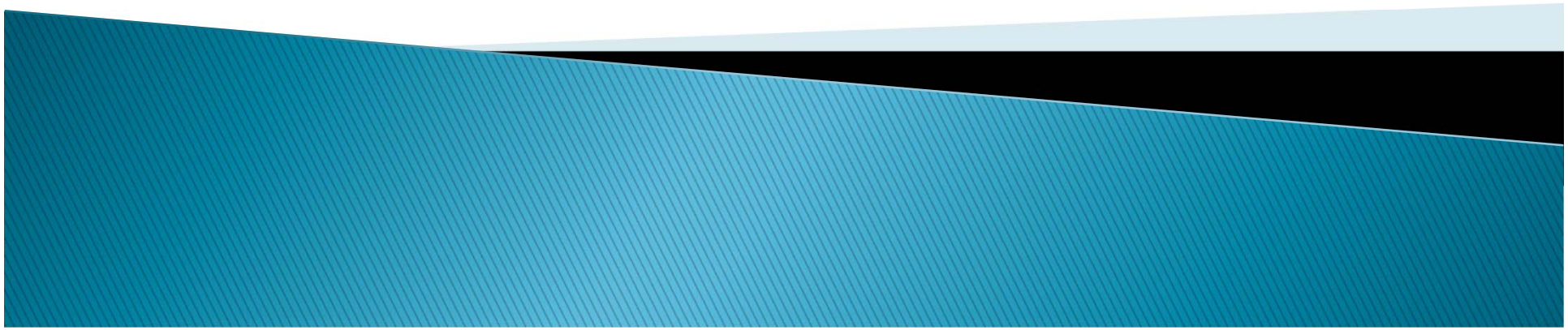
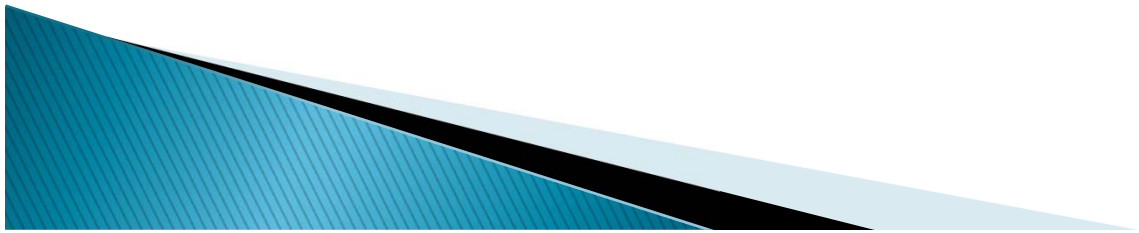
第六讲：字典



为什么需要字典？

- 现实世界中的事物有多方面的特征，例如一个学生有学号、姓名、性别、年级、专业等
- 软件系统是对现实世界的抽象，因为程序语言也需要提供机制来描述有多种特征的事物
- C语言提供了结构体struct，Python可以用更灵活的数据类型—字典



字典的声明和基本术语

- 语法：{key1: value1, key2: value2, ...}

```
1  stul = {'name': 'alice', 'sex': 'female', 'age': 19}
2  print(stul)
```

运行结果：

```
{'name': 'alice', 'sex': 'female', 'age': 19}
```

- key1, key2, ...称为键（key）
- value1, value2, ...称为值（value）
- key1: value1, key2: value2, ...称为项（item）
- 字典就是项的集合

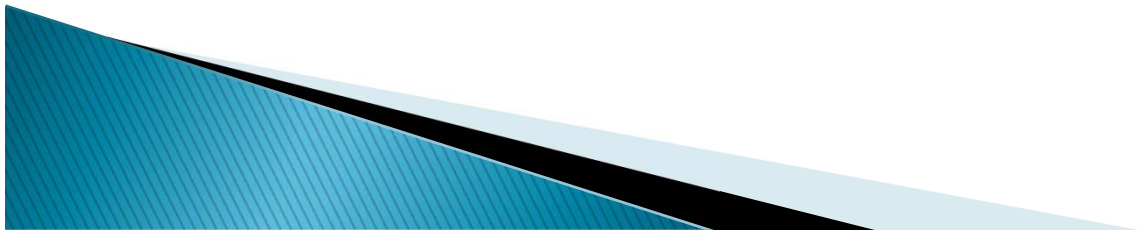
字典中的键和值

- 字典中的值没有任何限制，任何类型和值都可以作为字典中的值，字典中的值也是可以重复的
- 字典中的键不应该重复，在声明字典时如果键发生重复，只会保留一个

```
1    scores = {'alice': 80, 'bob': 92, 'carol': 92, 'alice': 75}
2    print(scores)
```

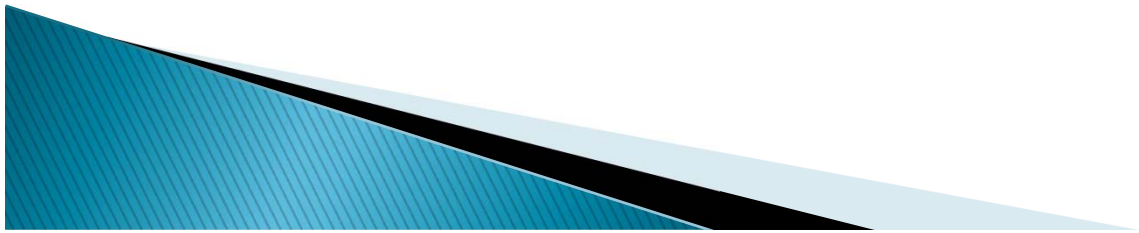
运行结果：

```
{'alice': 75, 'bob': 92, 'carol': 92}
```



字典中的键和值

- 并非所有类型都可以作为字典的键
- 可以作为键的类型：整数，浮点数，字符串（最常用），元组，函数等
- 不可以作为键的类型（原因后面再说）：列表，集合，字典等
- 一个字典可以同时包含多种不同类型的值，也可以同时包含多种不同类型的键



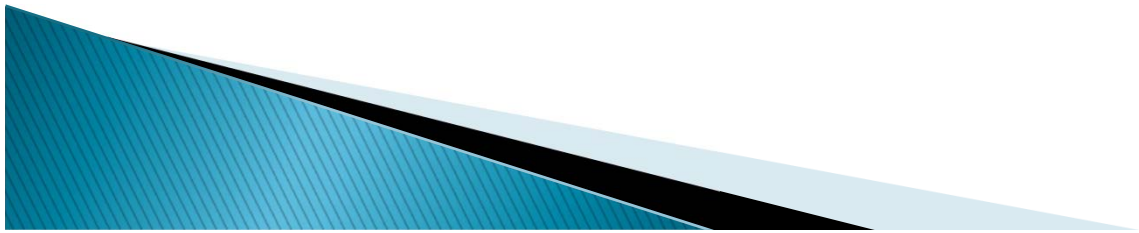
访问和修改字典的项

- 访问和修改都可以通过 字典名[键] 的方式进行

```
1    stul = {'name': 'alice', 'sex': 'female', 'age': 19}
2    print(stul['name'].title() + " is " + str(stul['age']) + " years old.")
3    stul['age'] = 20
4    print(stul['name'].title() + " is " + str(stul['age']) + " years old.")
```

运行结果：

```
Alice is 19 years old.
Alice is 20 years old.
```



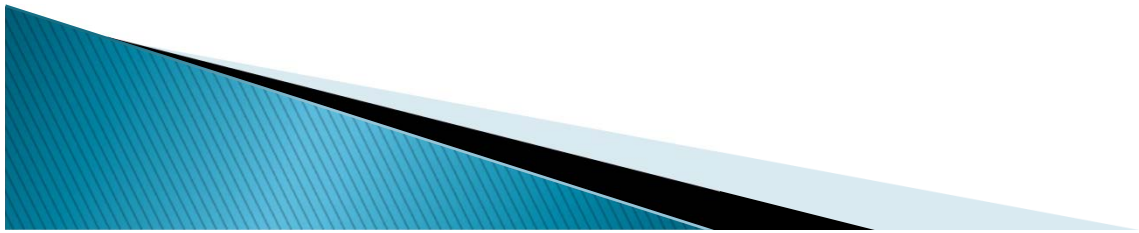
插入新的项

- 执行 字典名[键] = 值 时，如果键已经存在字典中，则执行修改操作，否则进行插入操作，也就是插入新的项

```
1  stul = {'name': 'alice', 'sex': 'female', 'age': 19}
2  print(stul)
3  stul['major'] = 'computer science'
4  print(stul)
```

运行结果：

```
{'name': 'alice', 'sex': 'female', 'age': 19}
{'name': 'alice', 'sex': 'female', 'age': 19, 'major': 'computer science'}
```



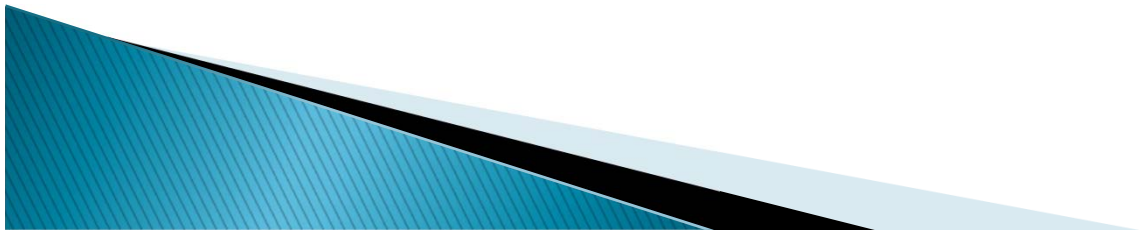
删除字典中的项

- 类似普通变量，我们可以用del 字典名[键]来删除指定键对应的项

```
1  stul = {'name': 'alice', 'sex': 'female', 'age': 19}
2  print(stul)
3  del stul['age']
4  print(stul)
```

运行结果：

```
{'name': 'alice', 'sex': 'female', 'age': 19}
{'name': 'alice', 'sex': 'female'}
```



遍历字典中的项

- 通过以下方式可以遍历字典中的所有项

  for 变量1, 变量2 in 字典名.items():

  ...

```
1   scores = {'alice': 80, 'bob': 92, 'carol': 92}
2   for name, score in scores.items():
3       print(name.title() + "'s score is " + str(score) + ".")
```

运行结果：

```
Alice's score is 80.
Bob's score is 92.
Carol's score is 92.
```

遍历字典中的键

- 通过以下方式可以遍历字典中的所有键

  for 变量 in 字典名.keys():

  ...

```
1  scores = {'alice': 80, 'bob': 92, 'carol': 92}
2  print("The score list contains the following students:")
3  for name in scores.keys():
4      print(name.title())
```

运行结果：

```
The score list contains the following students:
Alice
Bob
Carol
```

遍历字典中的值

- 通过以下方式可以遍历字典中的所有值

  for 变量 in 字典名.values():

  ...

```
1  scores = {'alice': 80, 'bob': 92, 'carol': 92}
2  print("The score list contains the following scores:")
3  for score in scores.values():
4      print(score)
```

运行结果：

```
The score list contains the following scores:
80
92
92
```

利用set来删除重复

- items(), keys(), values()这些方法返回的都是列表，因为列表元素是允许重复的，所以在上述例子用values()得到字典中所有值时是含有重复值的
- 可以通过set来将列表转换为集合（也是Python中的一种数据类型）

```
1  scores = {'alice': 80, 'bob': 92, 'carol': 92}
2  print("The score list contains the following scores:")
3  for score in set(scores.values()):
4      print(score)
```

运行结果：

```
The score list contains the following scores:
80
92
```

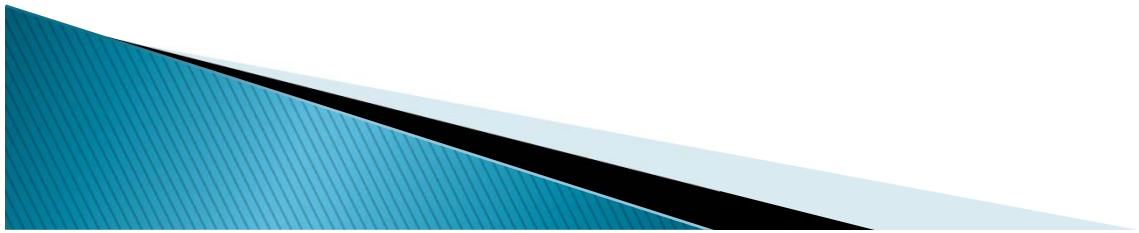
判断字典是否为空

- if后面紧跟字典名，可以判断这个字典是否为空

```
1  scores = {}
2  if scores:
3      print("It is not empty!")
4  else:
5      print("It is empty!")
```

运行结果：

```
It is empty!
```



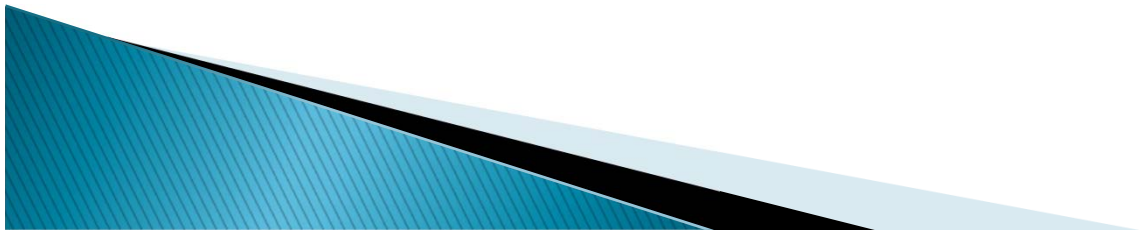
判断某个键是否存在

- 因为keys()方法返回字典中所有键组成的列表，所以可以利用这个方法来判断某个键是否存在

```
1   scores = {'alice': 80, 'bob': 92, 'carol': 92}
2   if 'tom' not in scores.keys():
3       print("Tom's score does not exist!");
```

运行结果：

```
Tom's score does not exist!
```



字典列表

- 列表的元素是多样的，如果列表中每个元素都是字典，我们称之为字典列表

```
1  stu1 = {'name': 'alice', 'sex': 'female', 'age': 19}
2  stu2 = {'name': 'bob', 'sex': 'male', 'age': 18}
3  stu3 = {'name': 'carol', 'sex': 'male', 'age': 20}
4  students = [stu1, stu2, stu3]
5  for student in students:
6      print(student['name'].title() + " is " + str(student['age']) + " years old.")
7
```

运行结果：

```
Alice is 19 years old.
Bob is 18 years old.
Carol is 20 years old.
```

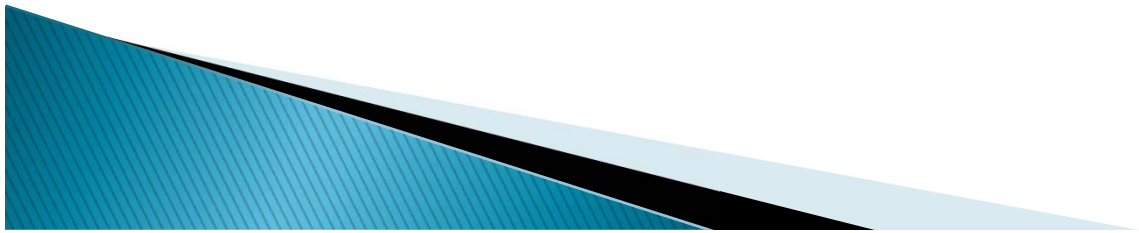
字典中包含列表

- 字典中的“值”可以是任何类型，因而也可以是列表

```
1  stu1 = {'name': 'alice', 'age': 19, 'scores': [100, 90, 87]}
2  stu2 = {'name': 'bob', 'age': 18, 'scores': [60, 85, 73]}
3  stu3 = {'name': 'carol', 'age': 20, 'scores': [95, 98, 88]}
4  students = [stu1, stu2, stu3]
5  for student in students:
6      print(student['name'].title() + "'s total score is " + str(sum(student['scores'])) + ".")
7
```

运行结果：

```
Alice's total score is 277.
Bob's total score is 218.
Carol's total score is 281.
```



字典中包含字典

- 字典中的“值”也可以是字典

```
1  stu1 = {'name': 'alice', 'scores': {'Math': 100, 'Chinese': 90, 'Chemistry': 87}}
2  stu2 = {'name': 'bob', 'scores': {'Networking': 60, 'Programming': 85, 'History': 73}}
3  stu3 = {'name': 'carol', 'scores': {'Math': 95, 'English': 98, 'Music': 88}}
4  students = [stu1, stu2, stu3]
5  for student in students:
6      print(student['name'].title() + "'s total score is " + str(sum(student['scores'].values())) + ".")
7
```

运行结果：

```
Alice's total score is 277.
Bob's total score is 218.
Carol's total score is 281.
```

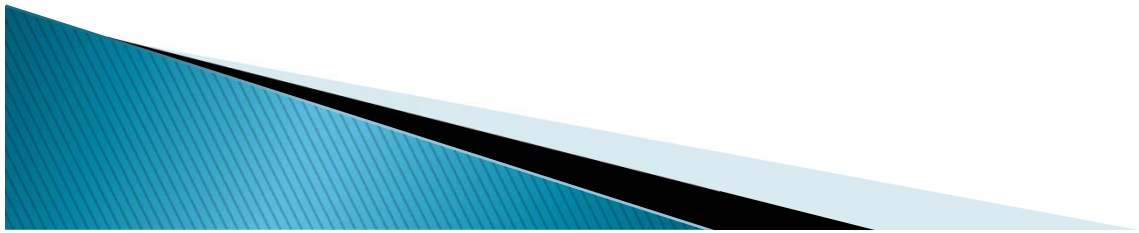
字典的赋值

- 和列表一样，用=号将一个字典变量赋值给另一个字典变量时，Python进行的是浅复制，也就是将两个变量指向同一个字典，因而修改其中一个会导致另一个也被修改

```
1    stu1 = {'name': 'alice', 'sex': 'female', 'age': 18}
2    stu2 = stu1
3    stu1['age'] = 19
4    print(stu1)
5    print(stu2)
```

运行结果：

```
{'name': 'alice', 'sex': 'female', 'age': 19}
{'name': 'alice', 'sex': 'female', 'age': 19}
```



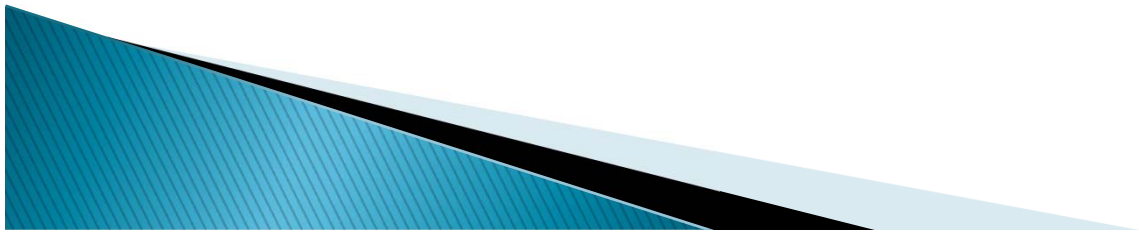
字典的赋值

- 一种避免浅复制的方法，是用dict来得到新的字典

```
1    stu1 = {'name': 'alice', 'sex': 'female', 'age': 18}
2    stu2 = dict(stu1)
3    stu1['age'] = 19
4    print(stu1)
5    print(stu2)
```

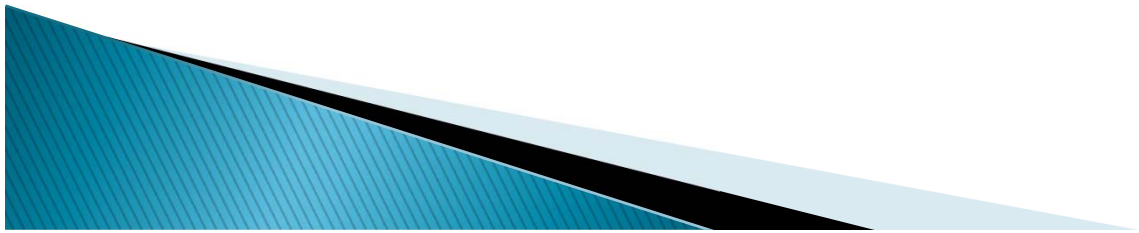
运行结果：

```
{'name': 'alice', 'sex': 'female', 'age': 19}
{'name': 'alice', 'sex': 'female', 'age': 18}
```



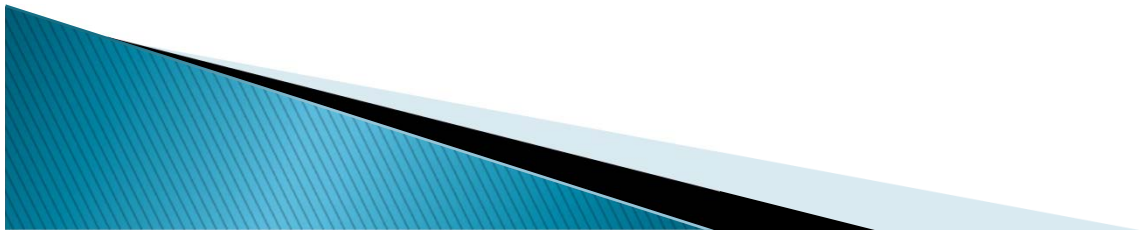
总结

- 字典的声明
- 字典项的访问、插入、删除和修改
- 遍历字典
- 字典和列表的相互嵌套



作业

- 教材中第6章课后的练习，选一些写到你的博客上



谢谢！