第四讲
字典、星号与模块
Lecture 4
Dictionary, Asterisk and Module

明玉瑞 Yurui Ming

yrming@gmail.com

声明
Disclaimer

- 本讲义在准备过程中由于时间所限，所用材料来源并未规范标示引用来源。所引材料仅用于教学所用，作者无意侵犯原著者之知识版权，所引材料之知识版权均归原著者所有；若原著者介意之，请联系作者更正及删除。

  The time limit during the preparation of these slides incurs the situation that not all the sources of the used materials (texts or images) are properly referenced or clearly manifested. However, all materials in these slides are solely for teaching and the author is with no intention to infringe the copyright bestowed on the original authors or manufacturers. All credits go to corresponding IP holders. Please address the author for any concern for remedy including deletion.

字典
Dictionary

- Python中的键值对是一对相互关联的值，键通过冒号关联到它的值。键的值可以是任何可以在 Python 中创建的对象，包括数字、字符串、列表及后面要讲的字典与类等。

  A key-value pair is a set of values associated with each other; every key is connected to its value by a colon. A key's value can be any object that one can create in Python, such as a number, a string, a list, or another dictionary or class covered later.

- Python中的字典由键值对组成的集合构成，即包裹在大括号{}内的由逗号分隔包含一系列键值对构成。可以使用键来访问与该键关联的值。理论上，可以在字典中存储任意数量的键值对。

  A dictionary in Python is a collection of key-value pairs, aka, a series of key-value pairs wrapped in braces, {}. One can use a key to access the value associated with that key. In theory dictionary can store as many key-value pairs.

字典
Dictionary

- 通过在字典名称后跟一组方括号，方括号内写键的名称，即可获取字典中存储的与键关联的值 。

  To get the value associated with a key, give the name of the dictionary and then place the key inside a set of square brackets.

- Python中的字典是动态结构，即随时可以将新的键值对添加到字典中。要添加一个新的键值对，可以在字典名称后跟的方括号中给出新键，后跟新值。 。

  Dictionaries are dynamic structures, new key-value pairs could be added into a dictionary at any time. To add a new key-value pair, give the name of the dictionary followed by the new key in square brackets along with the new value.

- 从 Python 3.7 开始，键值对在字典的存储顺序与它们添加到字典中的顺序相同。。

  As of Python 3.7, dictionaries retain the order of key-values pairs in storage in accordance with the sequence that they were added to the dictionary.

字典
Dictionary

- 有时从一个空字典开始，然后将每个新项逐渐添加到其中，会很方便。可以通过一对空的大括号定义一个空的字典。

  It’s sometimes convenient to start with an empty dictionary and then add each new item to it. An empty dictionary could be defined via an empty set of braces.

- 要修改字典中的值，可以在字典名称后跟的方括号中给出键名，然后是要与该键关联的新值。

  To modify a value in a dictionary, give the name of the dictionary with the key in square brackets and then the new value to be associated with that key.

- 可以使用 del 语句删除存储在字典中的不再需要的键值对，del 需要后跟字典名称与位于中括号中的要删除的键。

  del statement can be used to remove a no longer needed key-value pair stored in a dictionary. All del needs is the name of the dictionary and the key in the square brackets to be removed.

字典
Dictionary

- 使用方括号中写键的方式从字典中检索感兴趣的值可能会导致一个潜在的问题：如果要求的键不存在，则将收到错误消息。针对这种情况，可以使用 get() 方法设置一个默认值，如果请求的键不存在，将返回设定的值。get()方法第一个参数为指定查找的键，第二个可选参数设定为如果键不存在时，需要返回的值。

  Using keys in square brackets to retrieve the interested value from a dictionary might cause one potential problem: if the key doesn't exist, it will trigger an error. For this instance, one can use the get() method to set a default value that will be returned if the requested key doesn't exist. The get() method requires a key as a first argument. As a second optional argument, it is the value to be returned if the key doesn't exist.

```
>>> student = {"Name": "Andy", "ID": "6729120", "Literature": 90, "Math": 92}
>>> print(student)
{'Name': 'Andy', 'ID': '6729120', 'Literature': 90, 'Math': 92}
>>> print(f"{student['Name']} got {student['Math']} for Math")
Andy got 92 for Math
>>> print(f"{student['Name']} got {student['History']} for History")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'History'
>>> print(f"{student['Name']} got {student.get('History', 60)} for History")
Andy got 60 for History
```

字典
Dictionary

- 如果在get()的调用中未提供第二个参数并且键不存在，Python将返回值None。特殊值 None 表示“不存在的值”。这不是错误：它是一个特殊值，表示没有值。

  If the second argument in the call to get() are left out and the key doesn’t exist, Python will return the value None. The special value None means “no value exists.” This is not an error: it’s a special value meant to indicate the absence of a value..

- 某些情况下可能需要遍历字典，这种情况下基本有3种形式：
  - 遍历字典中所有元素，
  - 遍历所有键，
  - 遍历所有值。

  In some scenario we need to loop through the dictionary, there are basically three cases:
  - Looping through all key-value pairs,
  - Looping through all the keys in a dictionary,
  - Looping through all values in a dictionary.

字典
Dictionary

```
>>> student = {"Name": "Andy", "ID": "6729120", "Literature": 90, "Math": 92}
>>> for key, value in student.items():
...     print("{}: {}".format(key, value))
...
Name: Andy
ID: 6729120
Literature: 90
Math: 92
>>> for key in student.keys():
...     print(key)
...
Name
ID
Literature
Math
>>> for value in student.values():
...     print(value)
...
Andy
6729120
90
92
>>>
```

字典
Dictionary

- 有时可能会希望以不同的顺序循环字典。一种方法是对在for循环中返回的键进行排序。可以使用 sorted() 函数按顺序获取键的副本。

  Sometimes, one might need to loop through a dictionary in a different order. One way to do this is to sort the keys as they're returned in the for loop. one can use the sorted() function to get a copy of the keys in order.

```
>>> favorite_languages = {'jen': 'python', 'sarah': 'c', 'edward': 'ruby', 'phil': 'python', }
>>> for name, name_sorted in zip(favorite_languages.keys(), sorted(favorite_languages.keys())):
...     print(f"{name.title()}/{name_sorted.title()}, thank you for taking the poll.")
...
Jen/Edward, thank you for taking the poll.
Sarah/Jen, thank you for taking the poll.
Edward/Phil, thank you for taking the poll.
Phil/Sarah, thank you for taking the poll.
```

字典
Dictionary

- 有时可能会希望循环字典时，输出的键或值不要重复。此时可以用集合。 集合中的每个项目都必须是唯一的。

  Sometimes it may be desirable to loop over a dictionary so that the output keys or values do not repeat. set can be used in this instance. Each item in the set must be unique.

```
>>> favorite_languages = {'jen': 'python', 'sarah': 'c', 'edward': 'ruby', 'phil': 'python',}
>>> print("The following languages have been mentioned:")
The following languages have been mentioned:
>>> for language in set(favorite_languages.values()):
...     print(language.title())
...
Ruby
Python
C
>>>
```