Introduction To Python

Dr.Hajialiasgari

Tehran University
Of
Medical Science

November 8, 2024

TEHRAN UNIVERSITY
OF
MEDICAL SCIENCES

1. Python Dictionaries

A Story of Two Collections..

- `List:` A linear collection of values Lookup by position 0 .. length-1
- `Dictionary:` A linear collection of key-value pairs Lookup by "tag" or "key"

Dictionaries

- Dictionaries are Pythons most powerful data collection
- Dictionaries allow us to do fast database-like operations in Python
- Similar concepts in different programming languages
- We insert values into a `Dictionary` using a key and retrieve them using a key

Dictionary Example

```
>>> Wizard = dict()
>>> Wizard['Name'] = 'Harry'
>>> Wizard['Last Name'] = 'Potter'
>>> Wizard['Age'] = 12
>>> print(Wizard)
{'Name': 'Harry', 'Last Name': 'Potter', 'Age': 12}
>>> print(Wizard['Name'])
Harry
>>> Wizard['Age'] = Wizard['Age'] + 2
>>> print(Wizard)
{'Name': 'Harry', 'Last Name': 'Potter', 'Age': 14}
```

Comparing Lists and Dictionaries

- Dictionaries are like lists except that they use keys instead of positions to look up values

Code Example

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]

>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'age': 21, 'course': 182}
>>> ddd['age'] = 23
>>> print(ddd)
{'age': 23, 'course': 182}
```

Dictionary Literals (Constants)

- Dictionary literals use curly braces and have key : value pairs
- You can make an empty dictionary using empty curly braces

Code Example

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(jjj)
{'chuck': 1, 'fred': 42, 'jan': 100}
>>> ooo = { }
>>> print(ooo)
{}
>>>
```

Many Counters with a Dictionary

- One common use of dictionaries is counting how often we see something

```
>>> ccc = dict()
>>> ccc['csev'] = 1
>>> ccc['cwen'] = 1
>>> print(ccc)
{'csev': 1, 'cwen': 1}
>>> ccc['cwen'] = ccc['cwen'] + 1
>>> print(ccc)
{'csev': 1, 'cwen': 2}
```

Dictionary Tracebacks

- It is an error to reference a key which is not in the dictionary
- We can use the `in` operator to see if a key is in the dictionary

Code Example

```
>>> ccc = dict()
>>> print(ccc['csev'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'csev'
>>> 'csev' in ccc
False
```

When We See a New Name

- When we encounter a new name, we need to add a new entry in the dictionary and if this the second or later time we have seen the name, we simply add one to the count in the dictionary under that name

Code Example

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    if name not in counts:
       counts[name] = 1
    else :
        counts[name] = counts[name] + 1
print(counts)
```

The get Method for Dictionaries

- The pattern of checking to see if a key is already in a dictionary and assuming a default value if the key is not there is so common that there is a method called `get()` that does this for us
- We can use `get()` and provide a default value of zero when the key is not yet in the dictionary - and then just add one

get() Example

```
1 counts = dict()
2 names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
3 for name in names :
4     counts[name] = counts.get(name, 0) + 1
5 print(counts)
6
7 {'csev': 2, 'cwen': 2 , 'zqian': 1}
```

Definite Loops and Dictionaries

- We can write a `for`loop that goes through all the entries in a dictionary - actually it goes through all of the keys in the dictionary and looks up the values

Code Example

```
>>> counts = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> for key in counts:
...     print(key, counts[key])
...
chuck 1
fred 42
jan 100
>>>
```

Retrieving Lists of Keys and Values

- You can get a list of keys, values, or items (both) from a dictionary

Code Example

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(list(jjj))
['chuck', 'fred', 'jan']
>>> print(list(jjj.keys()))
['chuck', 'fred', 'jan']
>>> print(list(jjj.values()))
[1, 42, 100]
>>> print(list(jjj.items()))
[('chuck', 1), ('fred', 42), ('jan', 100)]
>>>
```

Bonus: Two Iteration Variables!

- We loop through the key-value pairs in a dictionary using *two* iteration variables
- Each iteration, the first variable is the key and the second variable is the corresponding value for the key

Code Example

```
1 jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
2 for aaa,bbb in jjj.items() :
3     print(aaa, bbb)
4
5 #output
6 chuck 1
7 fred 42
8 jan 100
```

End of Dictionaries