16-dict-I

March 20, 2022

1 Dictionaries

Dictionaries are a very useful kind of data-structure. It works more or less like lists, except that instead of indexing elements by a number (like `L[4]`), you can index using almost anything (strings, for example: `D["Qatar"]`). More precisely, a dictionary associates *keys* (the indices) with *values*. Keys are unique and are associated with only one value.

Another way of seeing a dictionary is as a set of key/value pairs. Note that there is no implicit order on the pairs, and the *keys are unique*.

The python documentation for dictionaries can be found in two links: - Introduction: https://docs.python.org/3/tutorial/datastructures.html#dictionaries - Dictionary functions: https://docs.python.org/3/library/stdtypes.html#typesmapping

1.1 Creating dictionaries

A concrete dictionary can be declared using curly braces and colons, like this:

[1]:
```
# Population of each country
pop = {"Qatar": 2881000,
       "US": 329500000,
       "Italy": 59550000,
       "China": 1402000000,
       "Iran": 83990000,
       "South Korea": 51780000,
       "Oman": 5107000,
       "Egypt": 102300000,
       "Jordan": 10200000,
       "Lebanon": 6825000,
       "Philipines": 109600000,
       "India": 1380000000
      }
```

The thing on the left of the colon is the key, and the thing on the right is the value.

Dictionaries can also be created from a list of pairs.

[2]:
```
L = [("Blue", 10), ("Red", 7), ("Green", 15), ("Black", 12)]
colors = dict(L)
print(colors)
```

```
{'Blue': 10, 'Red': 7, 'Green': 15, 'Black': 12}
```

And we can get a list of pairs in a dictionary using the `items` function and casting its result into a `list`:

[3]:
```
L = list(colors.items())
print(L)
```

```
[('Blue', 10), ('Red', 7), ('Green', 15), ('Black', 12)]
```

The **empty** dictionary is created either using the function `dict()` or empty curly braces `{}`.

[4]:
```
d1 = dict()
d2 = {}
print(d1)
print(d2)
```

```
{}
{}
```

1.2 Adding and modifying entries

Once you have a dictionary (possibly the empty one), you can add entries to it by assigning the value to the appropriate index:

[5]:
```
capitals = dict()
capitals["Qatar"] = "Doha"
capitals["Germany"] = "Berlin"
capitals["Vietnam"] = "Hanoi"
capitals["Nigeria"] = "Lagos"
print(capitals)
```

```
{'Qatar': 'Doha', 'Germany': 'Berlin', 'Vietnam': 'Hanoi', 'Nigeria': 'Lagos'}
```

If you assign a value to an existing key, the old value is **overwritten**.

[6]:
```
capitals["Nigeria"] = "Abuja"
print(capitals)
```

```
{'Qatar': 'Doha', 'Germany': 'Berlin', 'Vietnam': 'Hanoi', 'Nigeria': 'Abuja'}
```

1.3 Getting values

Values can be obtained by indexing with the key:

[7]:
```
v = capitals["Nigeria"]
print(v)
```

```
Abuja
```

If the key is not there, python raises a `KeyError`.

[20]:
```
v = capitals["Russia"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_1002424/3695980357.py in <module>
----> 1 v = capitals["Russia"]

KeyError: 'Russia'
```

To avoid `KeyError` errors, you can check if the key is in the dictionary before getting its value (see below), or you can use the `get()` function. This function is useful if you want to get a default value in case the key is not in the dictionary. The first parameter is the key whose value you want, the second one is the value that is going to be returned in case the key is not in the dictionary.

[9]:
```
v1 = pop.get("Qatar", 0)
print("Number of people in Qatar:", v1)

v2 = pop.get("Easter Island", 0)
print("Number of people in Easter Island:", v2)
```

```
Number of people in Qatar: 2881000
Number of people in Easter Island: 0
```

1.3.1 d.values()

Dictionaries provide us with the `values()` function, which returns an object containing all the values in the dictionary.

[10]:
```
vals = capitals.values()
print(vals)
```

```
dict_values(['Doha', 'Berlin', 'Hanoi', 'Abuja'])
```

To get the list inside this object, simply wrap it with `list()`:

[11]:
```
lvals = list(vals)
print(lvals)
```

```
['Doha', 'Berlin', 'Hanoi', 'Abuja']
```

1.4 Getting keys

There is no direct way to get one particular key from the dictionary (since they are not indexed like values).

To check if a *key* is in a dictionary, you can use `in`/`not in`. This is vey useful to avoid the `KeyError` shown above!

[12]:
```
"Qatar" in capitals
```

[12]: True

[13]:
```
"Jordan" not in capitals
```

[13]: True

1.4.1 d.keys()

We can get all keys in a dictionary at one via the function `keys()`. Similar to `values()`, it returns an object with all the dictionary keys. To get a list out of it, wrap it with `list()`.

[14]:
```
keys = list(capitals.keys())
print(keys)
```

```
['Qatar', 'Germany', 'Vietnam', 'Nigeria']
```

1.5 Number of entries

The function `len(d)` returns the number of entries in the dictionary.

[15]:
```
print(len(capitals))
```

```
4
```

1.6 Removing entries

An entry at key `k` can be removed from dictionary `d` via the `del` command:

[16]:
```
del pop["Oman"]
print(pop)
```

```
{'Qatar': 2881000, 'US': 329500000, 'Italy': 59550000, 'China': 1402000000,
'Iran': 83990000, 'South Korea': 51780000, 'Egypt': 102300000, 'Jordan':
10200000, 'Lebanon': 6825000, 'Philipines': 109600000, 'India': 1380000000}
```

1.7 Looping through dictionaries

We can loop through dictionaries using a `for` loop, where the loop variable will range among the dictionary's keys.

[17]:
```
for country in capitals:
        print("country =", country)
        if capitals[country] == "Hanoi":
            print(country + "'s capital is Hanoi")
```

```
country = Qatar
country = Germany
country = Vietnam
Vietnam's capital is Hanoi
country = Nigeria
```

1.8 Exercise 1

Ali recently got a 3D printer, and decided to open a business for printing messages in 3D letters. Printing in those printers is kind of slow, so he would like to group printing by letter. For example, if the message is `"Congratulations, Ahmad!"`, Ali would like to print 3 `"a"`s at once.

Help Ali figure out how much of each character he needs to print. Implement the function `charFreq(s)` that takes a string (the message) as a parameter, and returns a dictionary where the keys are characters, and values are the number of times Ali needs to print them. Remember that: - spaces do not need to be printed - capitalization matters (i.e. `"a"` is different from `"A"`) - punctuation needs to be printed

[18]:
```
def charFreq(s):
    return {}
```

1.9 Exercise 2

Suppose you have a dictionary `d` of population such as the one above, where the keys are countries and the values are the number of people in that country. Implement the function `sortByPop(d)` that returns a list of countries in decreasing order of population (so more populous countries first). You can assume no two countries have the same number of people.

[19]:
```
def sortByCases(d):
    return []
```