Dictionaries

تا اینجا ما درباره "توالی‌ها" در پایتون یاد گرفتیم، اما حالا قصد داریم به مبحث "نگاشت‌ها" در پایتون بپردازیم. اگر با زبان‌های برنامه‌نویسی دیگر آشنایی دارید، می‌توانید این دیکشنری‌ها را به عنوان جدول‌های هش در نظر بگیرید.

این بخش به عنوان یک مقدمه مختصر به دیکشنری‌ها خدمت می‌کند و شامل موارد زیر است:

1. ساختن یک دیکشنری
2. دسترسی به اشیاء در یک دیکشنری
3. تو در تو کردن دیکشنری‌ها
4. متدهای پایه‌ای دیکشنری

پس چه چیزی است نگاشت؟ نگاشت‌ها مجموعه‌ای از اشیاء هستند که توسط یک "کلید" ذخیره می‌شوند، به عکس توالی‌ها که اشیاء را بر اساس موقعیت نسبی ذخیره می‌کنند. این تفاوت مهم است، زیرا نگاشت‌ها ترتیب را حفظ نمی‌کنند زیرا اشیاء توسط یک کلید تعریف شده هستند.

یک دیکشنری پایتون از یک کلید و سپس یک مقدار مرتبط تشکیل شده است. این مقدار می‌تواند تقریباً هر شیء پایتونی باشد.

ساختن یک دیکشنری

بیایید ببینیم چگونه می‌توانیم دیکشنری‌ها را بسازیم تا درک بهتری از کارکرد آنها داشته باشیم!

برای تعریف دیکشینری ها سه روش وجود دارد

روش اول:

```
d = {
    key: value,
    key: value,
      .
      .
      .
    key: value
}
```

In [1]:

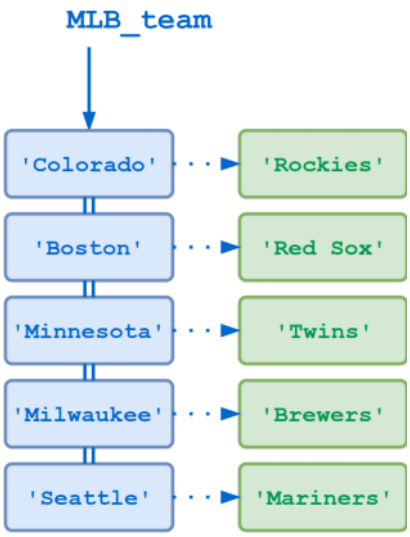
```
MLB_team = {
     'Colorado' : 'Rockies',
     'Boston'   : 'Red Sox',
     'Minnesota': 'Twins',
     'Milwaukee': 'Brewers',
     'Seattle'  : 'Mariners'
}
```

In [2]:

```
MLB_team
```

Out[2]:

```
{'Colorado': 'Rockies',
 'Boston': 'Red Sox',
 'Minnesota': 'Twins',
 'Milwaukee': 'Brewers',
 'Seattle': 'Mariners'}
```



روش دوم استفاده از تابع dict

In [ ]:

```
d = dict([
    (<key>, <value>),
    (<key>, <value),
      .
      .
      .
    (<key>, <value>)
])
```

In [3]:

```
MLB_team = dict([
     ('Colorado', 'Rockies'),
     ('Boston', 'Red Sox'),
     ('Minnesota', 'Twins'),
     ('Milwaukee', 'Brewers'),
     ('Seattle', 'Mariners')
])
```

In [4]:

```
MLB_team
```

Out[4]:

```
{'Colorado': 'Rockies',
 'Boston': 'Red Sox',
 'Minnesota': 'Twins',
 'Milwaukee': 'Brewers',
 'Seattle': 'Mariners'}
```

حالت استثنا روش دوم: در صورتیکه تمام کلیدها از جنس رشته باشد می توانیم بصورت خلاصه تر عمل کنیم:

In [5]:

```
MLB_team = dict(
     Colorado='Rockies',
     Boston='Red Sox',
     Minnesota='Twins',
     Milwaukee='Brewers',
     Seattle='Mariners'
)
```

In [6]:

```
MLB_team
```

Out[6]:

```
{'Colorado': 'Rockies',
 'Boston': 'Red Sox',
 'Minnesota': 'Twins',
 'Milwaukee': 'Brewers',
 'Seattle': 'Mariners'}
```

روش سوم را پس از یادگیری نحوه دسترسی به اعضای دیکشینری یاد میگیریم

نکته مهم: از پایتون 3.6 به بعد دیکشینری ها مانند لیست ها دارای ترتیب در اعضا می باشند ولی این ترتیب در مقایسه برابر بودن لحاظ نمی شود

In [7]:

```
l = [1,2,3]

l1 = [2,1,3]

l == l1
```

Out[7]:

```
False
```

In [8]:

```
d = {'k1': 'v1', 'k2': 'v2'}
d2 = {'k2': 'v2', 'k1': 'v1'}

d == d2
```

Out[8]:

```
True
```

دسترسی به مقادیر اعضا در دیکشینری

In [9]:

```
MLB_team
```

Out[9]:

```
{'Colorado': 'Rockies',
 'Boston': 'Red Sox',
 'Minnesota': 'Twins',
 'Milwaukee': 'Brewers',
 'Seattle': 'Mariners'}
```

In [10]:

```
MLB_team[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call las
t)
Cell In[10], line 1
----> 1 MLB_team[0]

KeyError: 0
```

In [11]:

```
MLB_team['Colorado']
```

Out[11]:

```
'Rockies'
```

In [12]:

```
MLB_team['Seattle']
```

Out[12]:

```
'Mariners'
```

In [13]:

```
1 MLB_team['Toronto']
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
Cell In[13], line 1
----> 1 MLB_team['Toronto']

KeyError: 'Toronto'
```

دیکشینری ها قابل تغییر هستند

In [14]:

```
1 MLB_team['Seattle'] = 'Seahawks'
```

In [15]:

```
1 MLB_team
```

Out[15]:

```
{'Colorado': 'Rockies',
 'Boston': 'Red Sox',
 'Minnesota': 'Twins',
 'Milwaukee': 'Brewers',
 'Seattle': 'Seahawks'}
```

In [16]:

```
1 MLB_team['Kansas City'] = 'Royals'
```

In [17]:

```
1 MLB_team
```

Out[17]:

```
{'Colorado': 'Rockies',
 'Boston': 'Red Sox',
 'Minnesota': 'Twins',
 'Milwaukee': 'Brewers',
 'Seattle': 'Seahawks',
 'Kansas City': 'Royals'}
```

این مهم است که توجه کنید دیکشنری‌ها در نوع داده‌هایی که می‌توانند نگه دارنده باشند، بسیار انعطاف پذیر هستند. به عنوان مثال:

In [18]:

```
my_dict = {'key1':123,'key2':[12,23,33],'key3':['item0','item1','item2']}
```

In [19]:

```
# Let's call items from the dictionary
my_dict['key3']
```

Out[19]:

```
['item0', 'item1', 'item2']
```

In [20]:

```
# Can call an index on that value
my_dict['key3'][0]
```

Out[20]:

```
'item0'
```

In [21]:

```
# Can then even call methods on that value
my_dict['key3'][0].upper()
```

Out[21]:

```
'ITEM0'
```

همانگونه که مشاهده نمودین تمام انواع داده ها می توانن بعنوان مقادیر یک کلید در دیکیشنری استفاده شود.

اما چه نوع داده هایی می توانند بعنوان کلید استفاده شوند؟

بصورت ساده می توان گفت که فقط داده های غیرقابل تغییر یا جهش می توانند بعنوان کلید استفاده شوند، یعنی اعداد، رشته ها، تاپل ها، بولین ها

اما دلیل اصلی این ویژگی، عدم امکان محاسبه مقدار hash برای انواع داده های قابل تغییر مانند لیست ها، دیکشینری ها و مجموعه ها می باشد

In [22]:

```
d = {0: 'v1', 5: 3.14}
```

In [23]:

```
d
```

Out[23]:

```
{0: 'v1', 5: 3.14}
```

In [24]:

```
d[0]
```

Out[24]:

```
'v1'
```

In [25]:

```
d = {(-1,-4): 'Tabriz', (0, 0): 'Tehran', (0, -3): 'Shiraz'}
```

In [26]:

```
d
```

Out[26]:

```
{(-1, -4): 'Tabriz', (0, 0): 'Tehran', (0, -3): 'Shiraz'}
```

In [27]:

```
d[(0, -3)]
```

Out[27]:

```
'Shiraz'
```

In [28]:

```
d = {[1,1]: 'not possible'}
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[28], line 1
----> 1 d = {[1,1]: 'not possible'}

TypeError: unhashable type: 'list'
```

روش سوم ساخت دیکشینری بصورت افزایشی می باشد

In [29]:

```
# Create a new dictionary
d = {}
```

In [30]:

```
# Create a new key through assignment
d['animal'] = 'Dog'
```

In [31]:

```
# Can do this with any object
d['answer'] = 42
```

In [32]:

```
#Show
d
```

Out[32]:

```
{'animal': 'Dog', 'answer': 42}
```

دیکشینری های تو در تو

امیدوارم شروع به درک قدرتمندی که پایتون با انعطاف‌پذیری در تو در تو کردن اشیاء و فراخوانی متدها در آن دارد کنید.
بیایید یک دیکشنری را درون یک دیکشنری ببینیم:

In [33]:

```
# Dictionary nested inside a dictionary nested inside a dictionary
d = {'key1':{'nestkey':{'subnestkey':'value'}}}
```

خیلی جالب هست! این یک تو در توی شگفت‌انگیز از دیکشنری‌هاست! بیایید ببینیم چگونه می‌توانیم آن مقدار را دریافت کنیم:

In [34]:

```
# Keep calling the keys
d['key1']['nestkey']['subnestkey']
```

Out[34]:

```
'value'
```

In [35]:

```
d['key1']['nestkey']['subnestkey'].upper()
```

Out[35]:

```
'VALUE'
```

برخی از متدهای دیکشینری ها

اینجا چندین روش برای متد یک دیکشنری وجود دارد. بیایید با چندتایی از آنها آشنا شویم:

In [36]:

```
# Create a typical dictionary
d = {'key1':1,'key2':2,'key3':3}
```

In [37]:

```
# Method to return a list of all keys
d.keys()
```

Out[37]:

```
dict_keys(['key1', 'key2', 'key3'])
```

In [38]:

```
# Method to grab all values
d.values()
```

Out[38]:

```
dict_values([1, 2, 3])
```

In [39]:

```
# Method to return tuples of all items  (we'll learn about tuples soon)
d.items()
```

Out[39]:

```
dict_items([('key1', 1), ('key2', 2), ('key3', 3)])
```

روش امن دریافت مقادیر یک کلید از دیکشینری با متد get

In [40]:

```
d['key5']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call las
t)
Cell In[40], line 1
----> 1 d['key5']

KeyError: 'key5'
```

In [41]:

```
d.get('key5')
```

In [42]:

```
d.get('key1')
```

Out[42]:

```
1
```

در دیکشینری ها ما دو نوع روش pop داریم.

In [43]:

```
# must give key
d.pop('key1')
```

Out[43]:

```
1
```

In [44]:

```
d
```

Out[44]:

```
{'key2': 2, 'key3': 3}
```

In [45]:

```
d.popitem()
```

Out[45]:

```
('key3', 3)
```

In [46]:

```
d
```

Out[46]:

```
{'key2': 2}
```

بروزرسانی یک دیکشینری با استفاده از اعضای دیکشینری دیگر با استفاده متد update

In [47]:

```
d1 = {'a' : 1, 'b': 5}

d2 = {'b': 777, 'c': -15}
```

In [48]:

```
d1.update(d2)
```

In [49]:

```
1 d1
```

Out[49]:

```
{'a': 1, 'b': 777, 'c': -15}
```

در این مباحث سعی شد تا با مفاهیم پایه و کاربردی دیکشینری ها در پایتون بخوبی آشنا شویم. در ادامه به بررسی نوع داده مجموعه و بولین ها خواهیم پرداخت.