30 Days Of Python: Day 8 - Dictionaries

LinkedIn | Follow @asabeneh
Author: Asabeneh Yetayeh
Second Edition: July, 2021

<< Day 7 | Day 9 >>

- Day 8
  - Dictionaries
    - Creating a Dictionary
    - Dictionary Length
    - Accessing Dictionary Items
    - Adding Items to a Dictionary
    - Modifying Items in a Dictionary
    - Checking Keys in a Dictionary
    - Removing Key and Value Pairs from a Dictionary
    - Changing Dictionary to a List of Items
    - Clearing a Dictionary
    - Deleting a Dictionary
    - Copy a Dictionary
    - Getting Dictionary Keys as a List
    - Getting Dictionary Values as a List
  - Exercises: Day 8

Day 8

Dictionaries

A dictionary is a collection of unordered, modifiable(mutable) paired (key: value) data type.

Creating a Dictionary

To create a dictionary we use curly brackets. () or the *dict()* built-in function.

```
# syntax
empty_dict = {}
# dictionary with data values
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
```

Example:

```
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_married':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
}
```

The dictionary above shows that a value could be any data types:string, boolean, list, tuple, set or a dictionary.

Dictionary Length

It checks the number of 'key: value' pairs in the dictionary.

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print(len(dct)) # 4
```

Example:

```
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_married':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
}
print(len(person)) # 7
```

Accessing Dictionary Items

We can access Dictionary items by referring to its key name.

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print(dct['key3']) # value3
print(dct['key4']) # value4
```

Example:

```
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_married':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
}
print(person['first_name'] # Asabeneh
print(person['country']    # Finland
print(person['skills'][-1] # Python
print(person['skills'][2]  # JavaScript
print(person['address']['street'] # Space street
print(person['city'])      # Error
```

Accessing an item by key name raises an error if the key does not exist. To avoid this error first we have to check if a key exist or we can use the *get* method. The get method returns None, which is a NoneType object data type, if the key does not exist.

```
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_married':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
}
print(person.get('first_name')) # Asabeneh
print(person.get('country'))    # Finland
print(person.get('skills')) #['HTML', 'CSS', 'JavaScript', 'React', 'Node', 'MongoDB', 'Python']
print(person.get('city'))       # None
```

Adding Items to a Dictionary

We can add new key and value pairs to a dictionary

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct['key5'] = 'value5'
```

Example:

```
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_married':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
}
person['first_name'] = 'Eyob'
person['age'] = 252
```

Checking Keys in a Dictionary

We use the *in* operator to check if a key exist in a dictionary

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print('key2' in dct) # True
print('key5' in dct) # False
```

Removing Key and Value Pairs from a Dictionary

- *pop(key)*: removes the item with the specified key name.
- *popitem()* removes the last item
- *del*: removes an item with specified key name

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct.pop('key3') # removes key3 item
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct.popitem() # removes the last item
del dct['key2'] # removes key2 item
```

Example:

```
person = {
    'first_name':'Asabeneh',
    'last_name':'Yetayeh',
    'age':250,
    'country':'Finland',
    'is_married':True,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
}

person.pop('first_name')      # Removes the firstname item
person.popitem()              # Removes the address item
del person['is_married']      # Removes the is_married item
```

Changing Dictionary to a List of Items

The *items()* method changes dictionary to a list of tuples.

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print(dct.items()) # dict_items([('key1', 'value1'), ('key2', 'value2'), ('key3', 'value3'), ('key4', 'value4')])
```

Clearing a Dictionary

If we don't want the items in a dictionary we can clear them using *clear()* method

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
print(dct.clear()) # None
```

Deleting a Dictionary

If we do not use the dictionary we can delete it completely

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
del dct
```

Copy a Dictionary

We can copy a dictionary using a *copy()* method. Using copy we can avoid mutation of the original dictionary.

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
dct_copy = dct.copy() # {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
```

Getting Dictionary Keys as a List

The *keys()* method gives us all the keys of a dictionary as a list.

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
keys = dct.keys()
print(keys)  # dict_keys(['key1', 'key2', 'key3', 'key4'])
```

Getting Dictionary Values as a List

The *values* method gives us all the values of a a dictionary as a list.

```
# syntax
dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}
values = dct.values()
print(values)    # dict_values(['value1', 'value2', 'value3', 'value4'])
```

🌕 You are astonishing. Now, you are super charged with the power of dictionaries. You have just completed day 8 challenges and you are 8 steps a head in to your way to greatness. Now do some exercises for your brain and muscles.

Exercises: Day 8

1. Create an empty dictionary called dog
2. Add name, color, breed, legs, age to the dog dictionary
3. Create a student dictionary and add first_name, last_name, gender, age, marital status, skills, country, city and address as keys for the dictionary
4. Get the length of the student dictionary
5. Get the value of skills and check the data type, it should be a list
6. Modify the skills values by adding one or two skills
7. Get the dictionary keys as a list
8. Get the dictionary values as a list
9. Change the dictionary to a list of tuples using *items()* method
10. Delete one of the items in the dictionary
11. Delete one of the dictionaries

🎉 CONGRATULATIONS ! 🎉

<< Day 7 | Day 9 >>