Dictionaries

Definitions

In this topic, we will examine the collection types which store **item pairs**. What does it mean?

Think of a real dictionary. It contains words and their meanings. In Python, you can accept the words as key and the meaning of the words as value.

A dictionary in Python is a collection of key-value pairs called **items** of a dictionary. The dictionary is enclosed by curly braces 👉🏻**{}**. Each pair (item) is separated by a comma and the key and value are separated by a colon.

## Dictionaries

You have earned 0 point(s) out of 0 point(s) thus far.

### Creating a Dictionary

A dictionary also can be created by enclosing pairs, separated by commas, in **curly-braces**. Looks like list or tuple, right?

And of course, we can use a function to create a dictionary : 'dict()' function. Let's create a simple empty dict :

empty\_dict\_1 = {}

empty\_dict\_2 = dict()

This is our first dict in this lesson. Now let's print its type.

input :

empty\_dict\_1 = {}

print(type(empty\_dict\_1))

output :

<class 'dict'>

The basic form of dict looks like :

my\_dict = {'key1': 'value1',

'key2': 'value2',

'key3': 'value3'

}

The syntax for accessing an **item** is very simple. We write a key that we want to access in square brackets. This method works both for adding items to a dict and for reading them from there.

In the following examples, you'll see several methods that allow us to create a dict and add a key-value pair to it.

input :

state\_capitals = {'Arkansas': 'Little Rock',

'Colorado': 'Denver',

'California': 'Sacramento',

'Georgia': 'Atlanta'

}

print(state\_capitals['Colorado']) # accessing method

output :

Denver

input :

state\_capitals = {'Arkansas': 'Little Rock',

'Colorado': 'Denver',

'California': 'Sacramento',

'Georgia': 'Atlanta'

}

state\_capitals['Virginia'] = 'Richmond' # adding a new item

print(state\_capitals)

output :

{'Arkansas': 'Little Rock',

'Colorado': 'Denver',

'California': 'Sacramento',

'Georgia': 'Atlanta',

'Virginia': 'Richmond'}

**💡Tips:**

* Note that values can be of different types.

mix\_dict = {'animal': ('dog', 'cat'), # tuple type

'planet': ['Neptun', 'Saturn', 'Jupiter'], # list type

'number': 40, # int type

'pi': 3.14, # float type

'is\_good': True} # bool type

And now, let's use dict() function to create a dictionary :

input :

dict\_by\_dict = dict(animal='dog', planet='neptun', number=40, pi=3.14,

    is\_good=True)

print(dict\_by\_dict)

output :

{'animal': 'dog',

'planet': 'neptun',

'number': 40,

'pi': 3.14,

'is\_good': True}

**⚠️Avoid ! :**

* Do not use quotes for keys when using the dict() function to create a dictionary.

**Q**: What is a dictionary in Python?  
**A**: Python dictionary is one of the supported data types in Python. It is an unordered collection of elements. The elements in dictionaries are stored as key–value pairs. Dictionaries are indexed by keys. For example, below we have a dict named my\_dict. It contains two keys, **fruit** and **vegatable**, along with their corresponding values, **banana** and **onion**.  
  
my\_dict = {'fruit':'banana', 'vegatable':'onion'}

**- Interview Q&A**

Main Operations with Dictionaries

There are several methods that allow us to access items, keys, and values. You can access all items using the **.items()** method, all keys using the **.keys()** method, and all values using the **.values()** method:

input :

dict\_by\_dict = {'animal': 'dog',

'planet': 'neptun',

'number': 40,

'pi': 3.14,

'is\_good': True}

print(dict\_by\_dict.items(), '\n')

print(dict\_by\_dict.keys(), '\n')

print(dict\_by\_dict.values())

output :

dict\_items([('animal', 'dog'), ('planet', 'neptun'),

('number', 40), ('pi', 3.14), ('is\_good', True)])

dict\_keys(['animal', 'planet', 'number', 'pi', 'is\_good'])

dict\_values(['dog', 'neptun', 40, 3.14, True])

You have learned that you can add a new item by assigning value to a key that is not in the dictionary. Likewise, you can add new items using the **.update()** method. Let's see :

input :

dict\_by\_dict = {'animal': 'dog',

'planet': 'neptun',

'number': 40,

'pi': 3.14,

'is\_good': True}

dict\_by\_dict.update({'is\_bad': False})

print(dict\_by\_dict)

output :

{'animal': 'dog',

'planet': 'neptun',

'number': 40,

'pi': 3.14,

'is\_good': True,

'is\_bad': False}

You can also remove an item using the **del** function:

**The formula syntax is : del dictionary\_name['key']**.

See the example.

input :

dict\_by\_dict = {'animal': 'dog',

'planet': 'neptun',

'number': 40,

'pi': 3.14,

'is\_good': True,

'is\_bad': False}

del dict\_by\_dict['animal']

print(dict\_by\_dict)

output :

{'planet': 'neptun',

'number': 40,

'pi': 3.14,

'is\_good': True,

'is\_bad': False}

Using the **in** and the **not in** operator, you can check if the key is in the dictionary.

* When we use the **in** operator; if the key is in the dictionary, the result will be True otherwise False.
* When we use the **not in**; if the key is not in the dictionary, the result will be True otherwise False.

Look at the example :

input :

dict\_by\_dict = {'planet': 'neptun',

'number': 40,

'pi': 3.14,

'is\_good': True,

'is\_bad': False}

print('pi' in dict\_by\_dict)

print('animal' not in dict\_by\_dict) # remember, we have deleted 'animal'

output :

True

True

Nested Dictionaries

In some cases, you need to work with the nested dict. When you decide to specialize in data science, we will work very often with dictionaries in the future.

school\_records={

"personal\_info":

{"kid":{"tom": {"class": "intermediate", "age": 10},

"sue": {"class": "elementary", "age": 8}

},

"teen":{"joseph":{"class": "college", "age": 19},

"marry":{"class": "high school", "age": 16}

},

},

"grades\_info":

{"kid":{"tom": {"math": 88, "speech": 69},

"sue": {"math": 90, "speech": 81}

},

"teen":{"joseph":{"coding": 80, "math": 89},

"marry":{"coding": 70, "math": 96}

},

},

}

We can use square brackets to access internal dicts :

input :

school\_records={

"personal\_info":

{"kid":{"tom": {"class":"intermediate", "age":10},

"sue": {"class":"elementary", "age":8}

},

"teen":{"joseph":{"class":"college", "age":19},

"marry":{"class":"high school", "age":16}

},

},

}

print(school\_records['personal\_info']['teen']['marry']['age'])

output :

16

| **nested_dictionary** |
| --- |
| *Diagram of Nested Dictionary* |

**💡Tips:**

* Dictionaries strongly resemble JSON syntax. The native json module in the Python standard library can be used to convert between JSON and dictionaries.

**✏️Homework:**

* What is '**JSON**' and what is it used for?

If you want to go deep into dicts, [**here**](https://docs.python.org/3.8/tutorial/datastructures.html#dictionaries) you will find what you want.









