

On dictionaries

support

Python Tutorial

- Python 3 - About
- Python 3 - Installation
- Python 3 - Environment
- Python 3 - Hello World
- Python 3 - Variables
- Python 3 - User Input
- Python 3 - Strings
- Python 3 - Lists
- Python 3 - Tuples
- Python 3 - Dictionary**
- Python 3 - Numbers
- Python 3 - Operators
- Python 3 - If Statements
- Python 3 - While Loops
- Python 3 - For Loops
- Python 3 - Functions
- Python 3 - Modules
- Python 3 - Classes
- Python 3 - Read/Write Files

Python Reference

- Python 3 - Operators
- Python 3 - Escape Sequences
- Python 3 - String Operators
- Python 3 - String Methods
- Python 3 - List Methods
- Python 3 - Numeric Operations
- Python 3 - Built-in Exceptions
- Python 3 - Exception Hierarchy

Python Dictionary

[← Python Tuples](#)[Python Numbers →](#)

In Python, a *dictionary* is an unordered collection of items, with each item consisting of a **key: value** pair (separated by a colon).

Create a Dictionary

You can create a dictionary by enclosing comma separated **key: value** pairs within curly braces `{}`. Like this:

```
d = {"Key1": "Value1", "Key2": "Value2"}
```

Here's an example of creating a dictionary, then printing it out, along with its type:

```
# Create the dictionary
planet_size = {"Earth": 40075, "Saturn": 378675, "Jupiter": 439264}
# Print the dictionary
print(planet_size)
# Print the type
print(type(planet_size))
```

RESULT

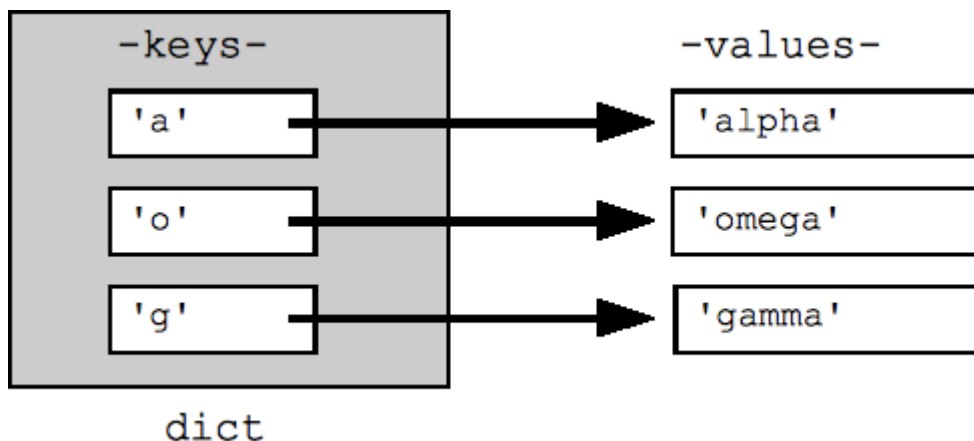
```
{'Earth': 40075, 'Saturn': 378675, 'Jupiter': 439264}
<class 'dict'>
```

But that's not the only way to create a dictionary. There's also a `dict()` function for creating dictionaries. And you can also use syntax variations within that function. Here are some examples:

<http://python-ds.com/python-dictionary>

```
planet_size = dict({"Earth": 40075, "Saturn": 378675, "Jupiter": 439264})
planet_size = dict([("Earth", 40075), ("Saturn", 378675), ("Jupiter", 439264)])
```

dictionaries



```
webstersDict = {'person': 'a human being',  
                'marathon': 'a running race that is about 26 miles',  
                'resist': 'to remain strong against the force',  
                'run': 'to move with haste; act quickly'}
```

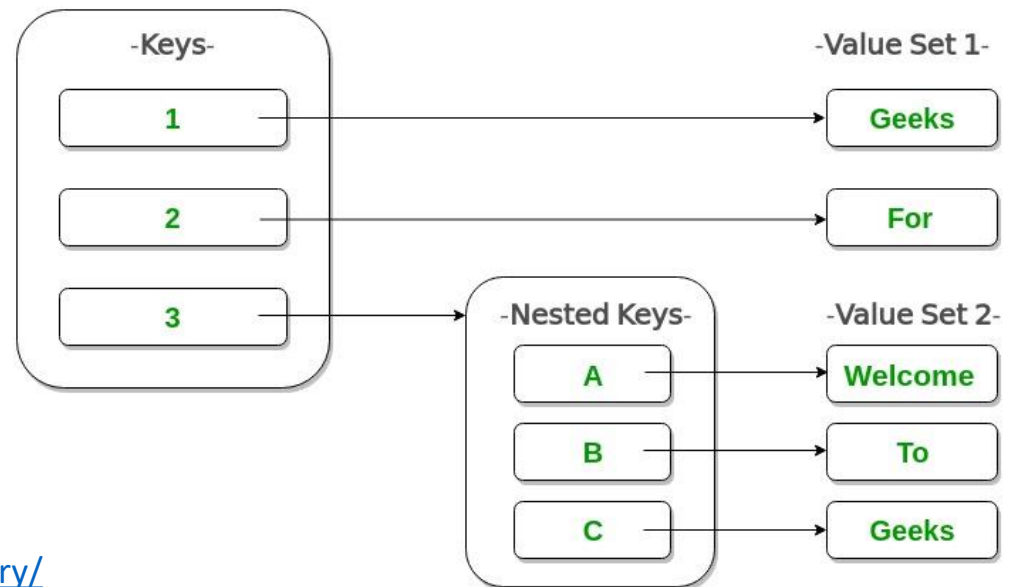
- dictionary key
- dictionary value

dictionaries: mixing...

```
# Creating a Nested Dictionary  
# as shown in the below image  
Dict = {1: 'Geeks', 2: 'For',  
        3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}  
  
print(Dict)
```

Output:

```
{1: 'Geeks', 2: 'For', 3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}
```



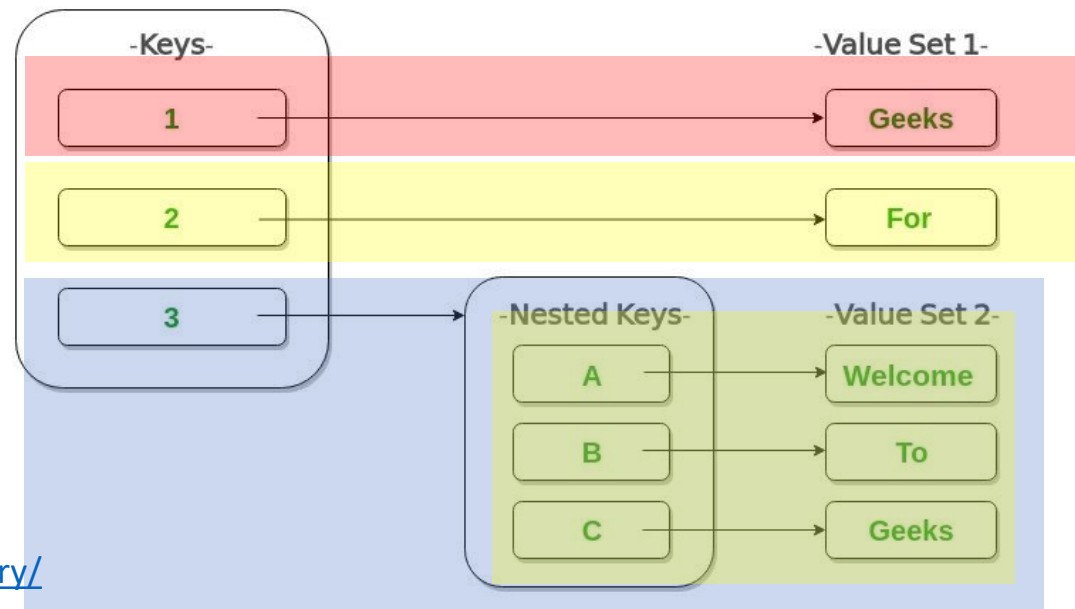
<https://www.geeksforgeeks.org/python-dictionary/>

dictionaries: mixing...

```
# Creating a Nested Dictionary  
# as shown in the below image  
Dict = {1: 'Geeks', 2: 'For',  
        3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}  
  
print(Dict)
```

Output:

```
{1: 'Geeks', 2: 'For', 3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}
```



<https://www.geeksforgeeks.org/python-dictionary/>

dictionaries: mixing...

```
dictionary_list = {'name': 'Ariel', 'hobbies': ['painting' , 'singing' , 'cooking']  
print ("dictionary_list['name']:", dictionary_list['name'])  
print ("dictionary_list['hobbies']:", dictionary_list['name'])  
//use index to access specific value  
print ("dictionary_list['hobbies'][0]:", dictionary_list['name'][0])  
print ("dictionary_list['hobbies'][1]:", dictionary_list['name'][1])  
print ("dictionary_list['hobbies'][2]:", dictionary_list['name'][2])
```

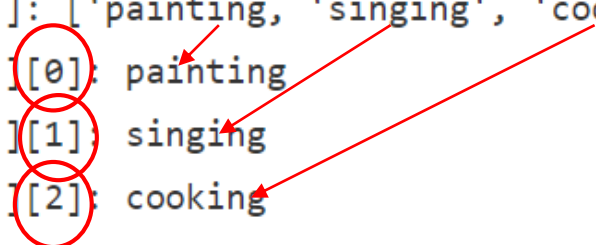
```
dictionary_list['name']: Ariel  
dictionary_list['hobbies']: ['painting', 'singing', 'cooking']  
dictionary_list['hobbies'][0]: painting  
dictionary_list['hobbies'][1]: singing  
dictionary_list['hobbies'][2]: cooking
```

<https://www.gangboard.com/blog/python-dictionary/>

dictionaries: mixing...

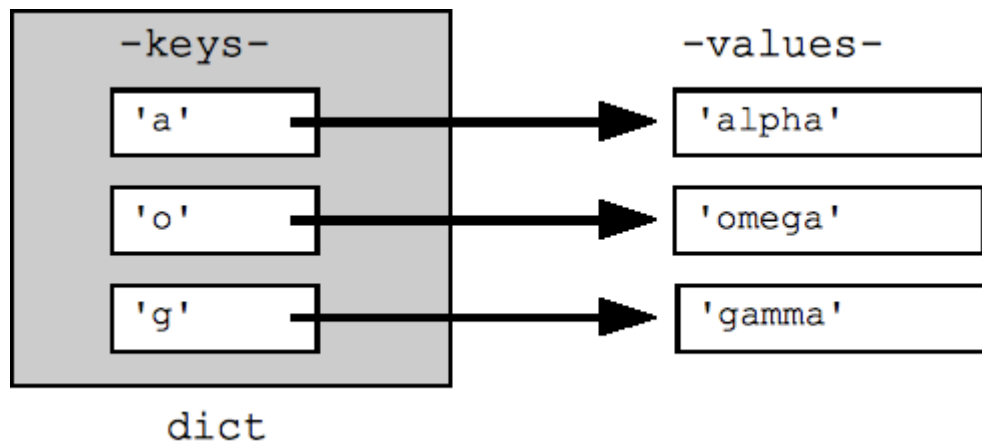
```
dictionary_list = {'name': 'Ariel', 'hobbies': ['painting' , 'singing' , 'cooking']  
print ("dictionary_list['name']:", dictionary_list['name'])  
print ("dictionary_list['hobbies']:", dictionary_list['name'])  
//use index to access specific value  
print ("dictionary_list['hobbies'][0]:", dictionary_list['name'][0])  
print ("dictionary_list['hobbies'][1]:", dictionary_list['name'][1])  
print ("dictionary_list['hobbies'][2]:", dictionary_list['name'][2])
```

```
dictionary_list['name']: Ariel  
dictionary_list['hobbies']: ['painting', 'singing', 'cooking']  
dictionary_list['hobbies'][0]: painting  
dictionary_list['hobbies'][1]: singing  
dictionary_list['hobbies'][2]: cooking
```



<https://www.gangboard.com/blog/python-dictionary/>

dictionaries_: keys, values & items



```
d.keys() → [ 'a', 'o', 'g' ]  
d.values() → [ 'alpha', 'omega', 'gamma' ]  
d.items() → [ 'a':'alpha', 'o':'omega', 'g':'gamma' ]  
d.count() → 3  
'a' in d → True  
'b' in d → False
```


Dictionaries: listing,

Using the key (implicit)

```
>>> for key in a_dict:
...     print(key, '->', a_dict[key])
...
color -> blue
fruit -> apple
pet -> dog
```

Using the key (explicit)

```
>>> for key in a_dict.keys():
...     print(key, '->', a_dict[key])
...
color -> blue
fruit -> apple
pet -> dog
```

Using the items → list of key,values

```
>>> for item in a_dict.items():
...     print(item)
...
('color', 'blue')
('fruit', 'apple')
('pet', 'dog')

>>> for key, value in a_dict.items():
...     print(key, '->', value)
...
color -> blue
fruit -> apple
pet -> dog
```

Using the values (explicit)

```
>>> for value in a_dict.values():
...     print(value)
...
blue
apple
dog
```

dictionaries: in?

```
>>> a_dict = {'color': 'blue', 'fruit': 'apple', 'pet': 'dog'}
>>> 'pet' in a_dict.keys()
True
>>> 'apple' in a_dict.values()
True
>>> 'onion' in a_dict.values()
False
```

<https://realpython.com/iterate-through-dictionary-python/>

dictionaries: delete & update

```
>>> prices = {'apple': 0.40, 'orange': 0.35, 'banana': 0.25}
>>> for k, v in prices.items():
...     prices[k] = round(v * 0.9, 2) # Apply a 10% discount
...
>>> prices
{'apple': 0.36, 'orange': 0.32, 'banana': 0.23}
```

```
>>> prices = {'apple': 0.40, 'orange': 0.35, 'banana': 0.25}
>>> for key in list(prices.keys()): # Use a list instead of a view
...     if key == 'orange':
...         del prices[key] # Delete a key from prices
...
>>> prices
{'apple': 0.4, 'banana': 0.25}
```

<https://realpython.com/iterate-through-dictionary-python/>



Dictionaries misc

```
webstersDict
```

```
{'marathon': '26 mile race',
 'person': 'a human being',
 'run': 'to move with haste; act quickly',
 'shoe': 'an external covering for the human foot'}
```

```
webstersDict.update({'ran': 'past tense of run',
                    'shoes': 'plural of shoe'})
```

```
webstersDict
```

```
{'marathon': '26 mile race',
 'person': 'a human being',
 'ran': 'past tense of run',
 'run': 'to move with haste; act quickly',
 'shoe': 'an external covering for the human foot',
 'shoes': 'plural of shoe'}
```

```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5, 'the': 90}
```

```
storyCount.pop('the')
```

```
90
```

```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5}
```

```
print(storyCount.get('Michael'))
```

```
12
```

<https://medium.com/@GalarnykMichael/python-basics-10-dictionaries-and-dictionary-methods-4e9efa70f5b9>



Dictionaries misc

```
student_dictionary = {'name' : 'Lisa', 'age' : 6, 'grade' : '1' }  
student_dictionary.pop('grade')  
print (student_dictionary)
```

Output:

```
{'name': 'Lisa', 'age': 6 }
```

```
dict = {1: "one", 2: "three"}  
dict_update = {2: "two"}  
#value of key 2 is updated  
dict.update(dict_update)  
print(dict)
```

Output:

```
{1: 'one', 2: 'two'}
```

```
dog = { "breed": "labrador", "color": "dusty white", "sex": "female" }  
x = dog.values()  
print(x)
```

Output:

```
dict_values(['labrador', 'dusty white', 'female'])
```

<https://www.gangboard.com/blog/python-dictionary/>

Dictionary functions

METHODS	DESCRIPTION
<code>copy()</code>	They <code>copy()</code> method returns a shallow copy of the dictionary.
<code>clear()</code>	The <code>clear()</code> method removes all items from the dictionary.
<code>pop()</code>	Removes and returns an element from a dictionary having the given key.
<code>popitem()</code>	Removes the arbitrary key-value pair from the dictionary and returns it as tuple.
<code>get()</code>	It is a conventional method to access a value for a key.
<code>dictionary_name.values()</code>	returns a list of all the values available in a given dictionary.
<code>str()</code>	Produces a printable string representation of a dictionary.
<code>update()</code>	Adds dictionary dict2's key-values pairs to dict
<code>setdefault()</code>	Set dict[key]=default if key is not already in dict
<code>keys()</code>	Returns list of dictionary dict's keys
<code>items()</code>	Returns a list of dict's (key, value) tuple pairs
<code>has_key()</code>	Returns true if key in dictionary dict, false otherwise
<code>fromkeys()</code>	Create a new dictionary with keys from seq and values set to value.
<code>type()</code>	Returns the type of the passed variable.
<code>cmp()</code>	Compares elements of both dict.

<https://www.geeksforgeeks.org/python-dictionary/>

Function with Description	Methods with Description
<code>cmp(dict1, dict2)</code> ⓘ Compares elements of both dict.	<code>dict.clear()</code> ⓘ Removes all elements of dictionary <i>dict</i>
<code>len(dict)</code> ⓘ Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.	<code>dict.copy()</code> ⓘ Returns a shallow copy of dictionary <i>dict</i>
<code>str(dict)</code> ⓘ Produces a printable string representation of a dictionary	<code>dict.fromkeys()</code> ⓘ Create a new dictionary with keys from seq and values set to <i>value</i> .
<code>type(variable)</code> ⓘ Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.	<code>dict.get(key, default=None)</code> ⓘ For <i>key</i> key, returns value or default if key not in dictionary
	<code>dict.has_key(key)</code> ⓘ Returns <i>true</i> if key in dictionary <i>dict</i> , <i>false</i> otherwise
	<code>dict.items()</code> ⓘ Returns a list of <i>dict</i> 's (key, value) tuple pairs
	<code>dict.keys()</code> ⓘ Returns list of dictionary <i>dict</i> 's keys
	<code>dict.setdefault(key, default=None)</code> ⓘ Similar to <i>get()</i> , but will set <i>dict[key]=default</i> if <i>key</i> is not already in <i>dict</i>
	<code>dict.update(dict2)</code> ⓘ Adds dictionary <i>dict2</i> 's key-values pairs to <i>dict</i>
	<code>dict.values()</code> ⓘ Returns list of dictionary <i>dict</i> 's values

Dictionary: when key does not exist

- Getting a non existent key provokes error

- KeyError

```
country_dict = {'India' : 'IN', 'Australia' : 'AU', 'Brazil' : 'BR'}  
print(country_dict['Australia'])  
print(country_dict['Canada']) # This will return error
```

- Solution

- Handle error
 - **Default value (the second parameter)**

<https://www.tutorialspoint.com/handling-missing-keys-in-python-dictionaries>
<https://www.geeksforgeeks.org/handling-missing-keys-python-dictionaries/>



Dictionary: default value

```
print(storyCount.get('chicken'))
```

None

```
print(storyCount['chicken'])
```

```
-----
KeyError                                Traceback (most recent call last)
<ipython-input-20-27ef036ded95> in <module>()
----> 1 print(storyCount['chicken'])
```

KeyError: 'chicken'

```
print(storyCount.get('chicken', 0))
```

0

```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5, 'the': 90}
```

```
storyCount.pop('the')
```

90

```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5}
```

```
print(storyCount.get('Michael'))
```

12

```
country_dict = {'India' : 'IN', 'Australia' : 'AU', 'Brazil' : 'BR'}
print(country_dict.get('Australia', 'Not Found'))
print(country_dict.get('Canada', 'Not Found'))
```

<https://www.tutorialspoint.com/handling-missing-keys-in-python-dictionaries>
<https://www.geeksforgeeks.org/handling-missing-keys-python-dictionaries/>



Dictionaries curiosities

```
sequence_keys = {'A', 'B', 'AB', 'O' }  
value = 'blood type'  
bloodtype = dict.fromkeys(sequence_keys, value)  
print (bloodtype)
```

```
{'A': 'blood type', 'B': 'blood type', 'AB': 'blood type', 'O': 'blood type'}
```

Similar to

```
bloodtype = {}  
Value = 'blood type'  
for k in sequence_keys:  
    bloodtype[k]=value
```

<https://www.gangboard.com/blog/python-dictionary/>

The END