Dictionaries

Dictionaries

dic·tion·ar·y ˈdikSHəˌnerē/

noun

a book or electronic resource that lists the words of a language (typically in alphabetical order) and gives their meaning, or gives the equivalent words in a different language, often also providing information about pronunciation, origin, and usage.

Dictionaries

A dictionary with three entries:

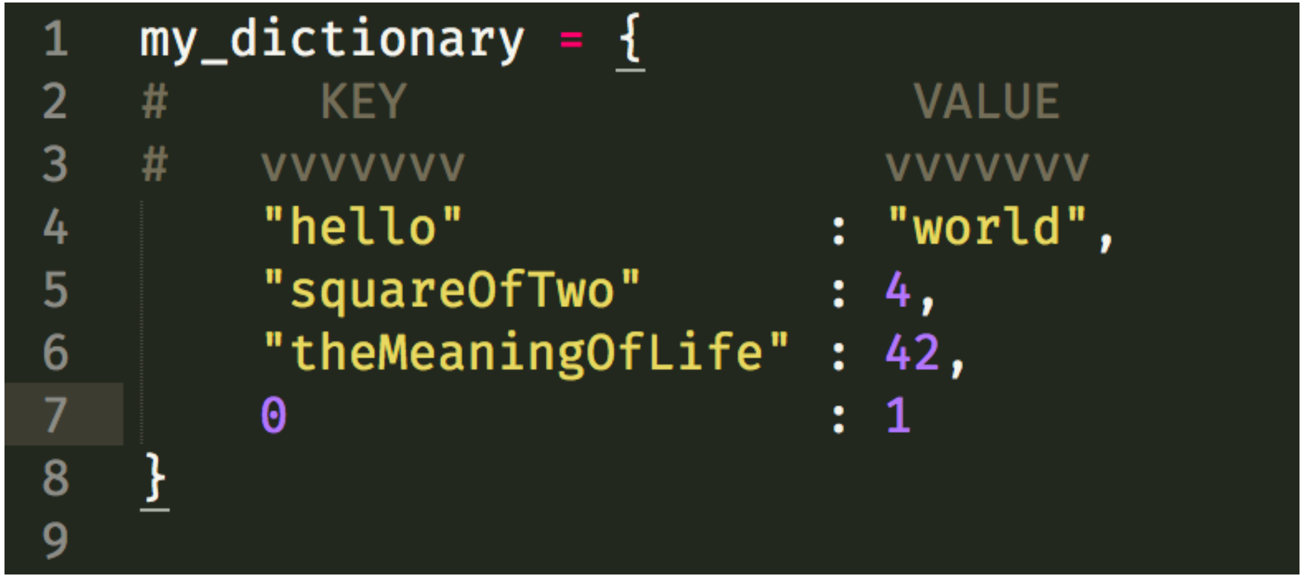
```python
oxford = {
    'haimish': 'homey; cozy and unpretentious.',
    'bon mot': 'a witty remark or comment; clever saying; witticism.',
    'albatross': 'something burdensome that impedes action or progress.'
}
```

An empty dictionary:

```python
empty_dictionary = {}
```

Dictionaries

Keys can be any primitive type and values can be *anything*.

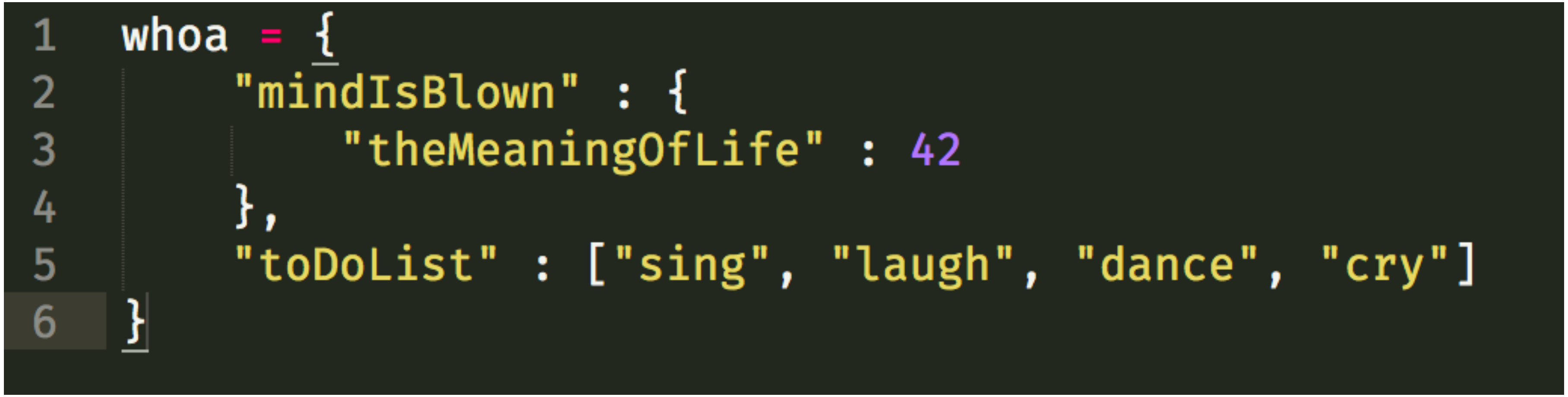


```
my_dictionary = {
#       KEY                      VALUE
#     vvvvvvv                   vvvvvvv
      "hello"                : "world",
      "squareOfTwo"          : 4,
      "theMeaningOfLife"     : 42,
      0                      : 1
}

```

Dictionaries

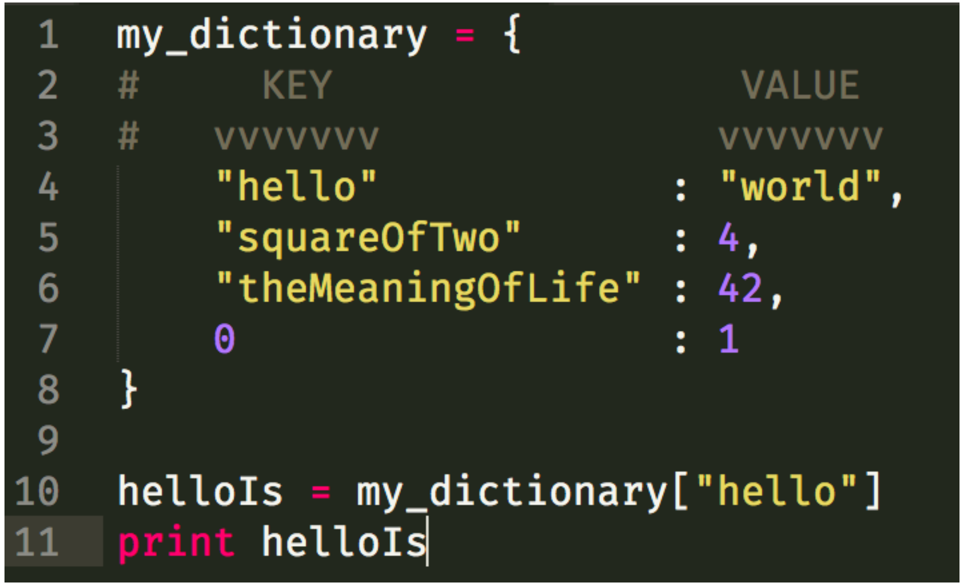
Even other dictionaries…



```
whoa = {
        "mindIsBlown" : {
                "theMeaningOfLife" : 42
        },
        "toDoList" : ["sing", "laugh", "dance", "cry"]
}
```

Dictionaries

So, how do you get stuff from dictionaries? Easy. You *index* the dictionary by the *key you're looking for.*

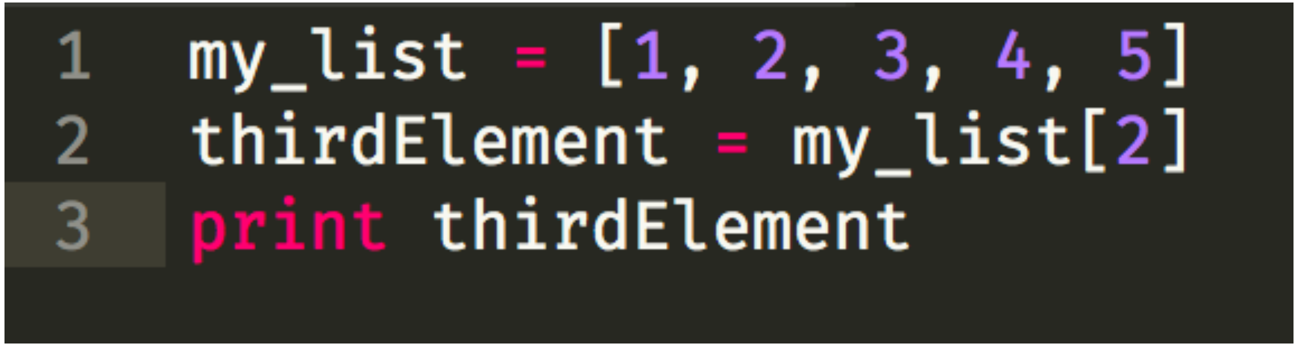


```
my_dictionary = {
#       KEY                       VALUE
#    vvvvvvv                     vvvvvvv
     "hello"                   : "world",
     "squareOfTwo"             : 4,
     "theMeaningOfLife"        : 42,
     0                         : 1
}

helloIs = my_dictionary["hello"]
print helloIs
```

Dictionaries

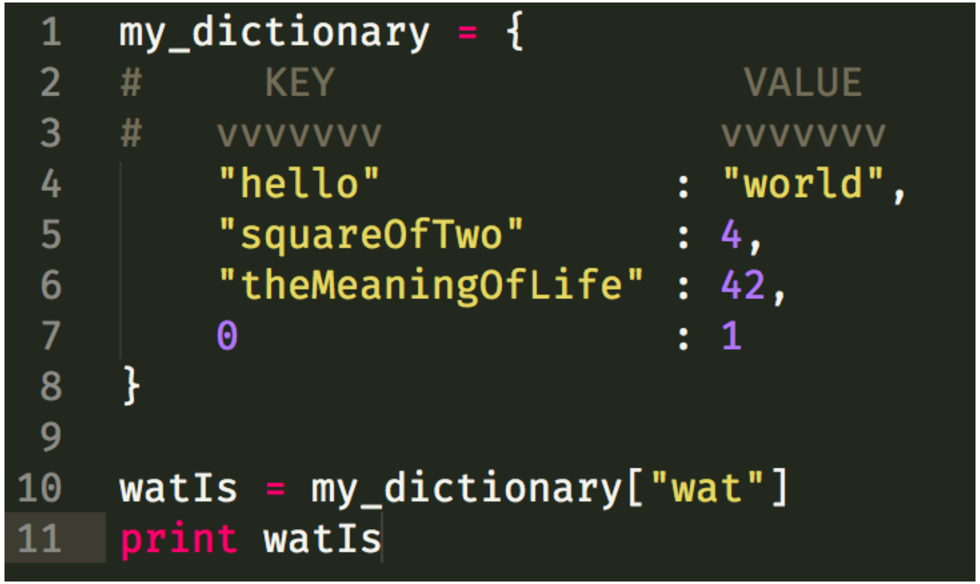
Look familiar? Yeah! That’s exactly how we index lists, except with a list the index can only be numbers.



```
my_list = [1, 2, 3, 4, 5]
thirdElement = my_list[2]
print thirdElement
```

Dictionaries

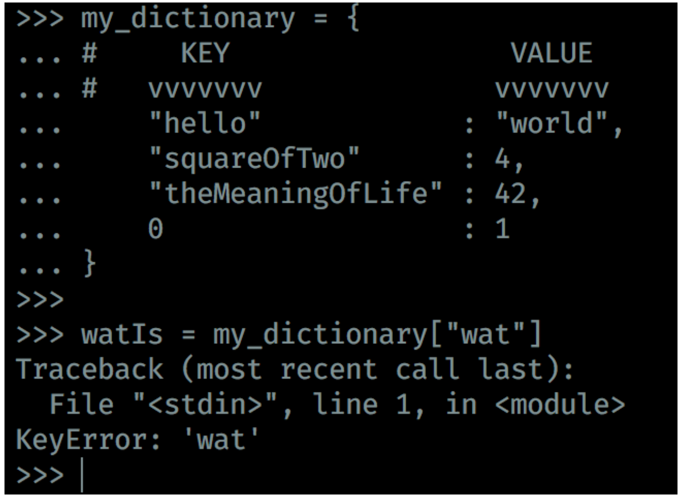
So, what happens if we try to access a dictionary with a key that doesn't exist?



```
my_dictionary = {
#        KEY                              VALUE
#     vvvvvvv                           vvvvvvv
      "hello"                         : "world",
      "squareOfTwo"                   : 4,
      "theMeaningOfLife"              : 42,
      0                               : 1
}

watIs = my_dictionary["wat"]
print watIs
```

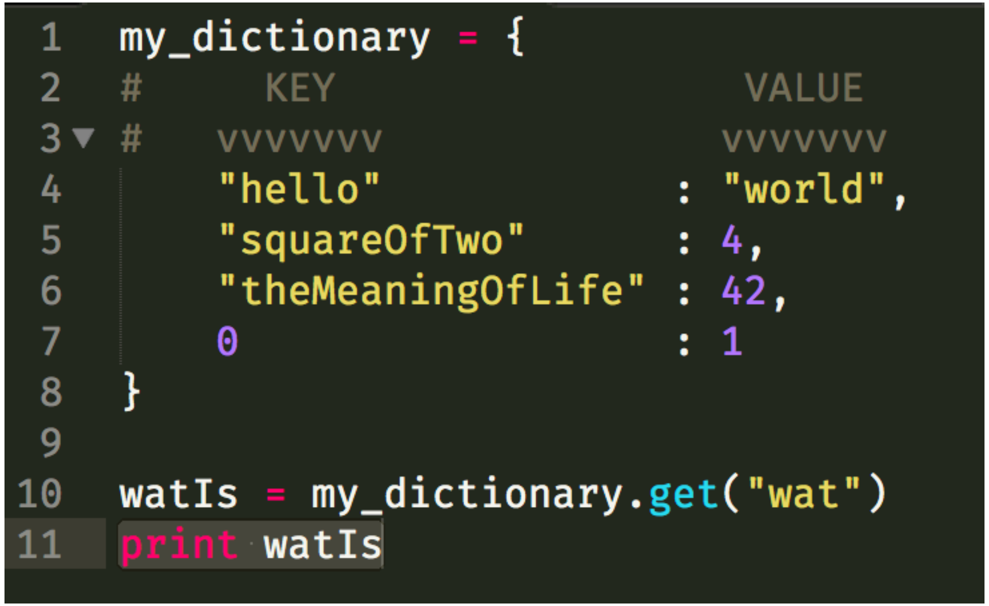
Dictionaries



```
>>> my_dictionary = {
... #     KEY                   VALUE
... #   vvvvvvv                vvvvvvv
...     "hello"              : "world",
...     "squareOfTwo"        : 4,
...     "theMeaningOfLife"   : 42,
...     0                    : 1
... }
>>>
>>> watIs = my_dictionary["wat"]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'wat'
>>> 
```

Dictionaries

But that's ok, there are ways to avoid this problem (and you should avoid it). Good Practice: Only index keys that you know to exist.



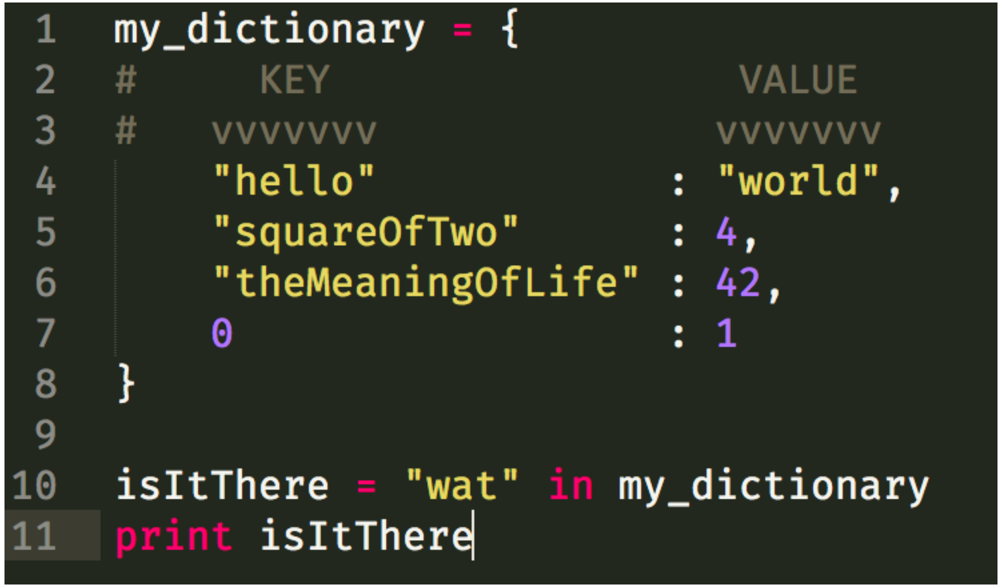
```
my_dictionary = {
#      KEY                   VALUE
#    vvvvvvv                vvvvvvv
     "hello"              : "world",
     "squareOfTwo"        : 4,
     "theMeaningOfLife"   : 42,
     0                    : 1
}

watIs = my_dictionary.get("wat")
print watIs
```

Use the `get()` method to safely retrieve dictionary items where you're not sure if the key exists. It will return `None` if the key is not in the dictionary.

Dictionaries

Or, you could check first. There's times where this is appropriate instead of looking for default values. Who knows what the value is?



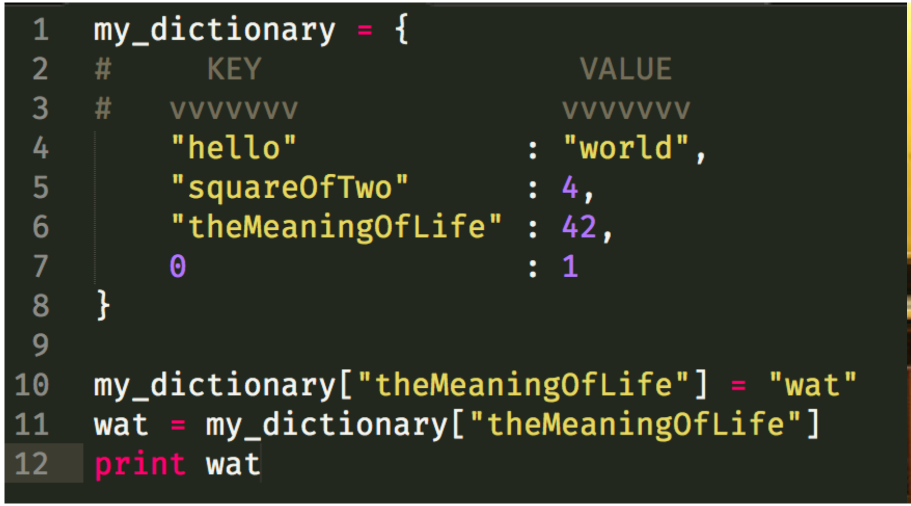
```
my_dictionary = {
#       KEY                      VALUE
#     vvvvvvv                   vvvvvvv
      "hello"               : "world",
      "squareOfTwo"         : 4,
      "theMeaningOfLife"    : 42,
      0                     : 1
}

isItThere = "wat" in my_dictionary
print isItThere
```

Use the `in` operator similar to how you loop through lists. It makes sense in english here; you're checking to see if the key is **in** the dictionary!

Dictionaries

Setting a value is like putting something in a mailbox.

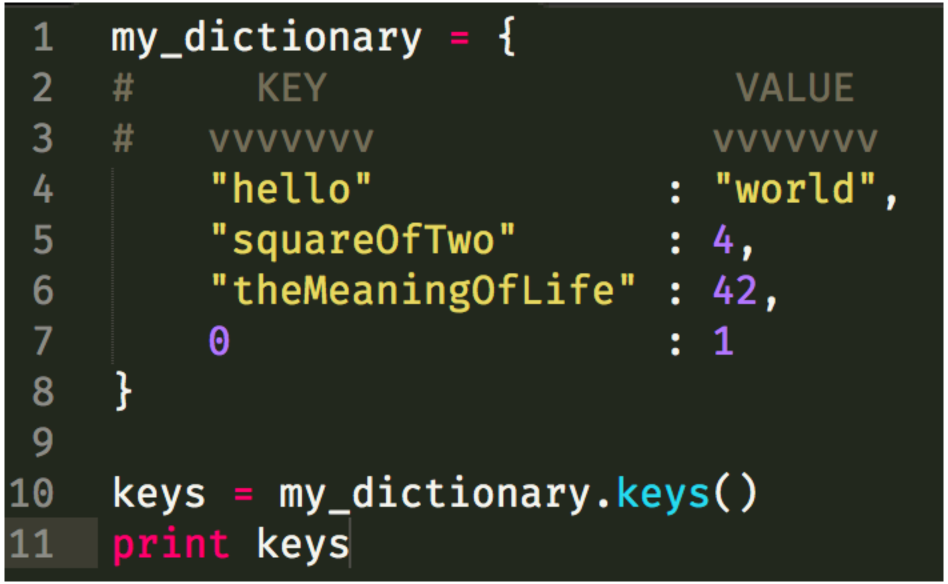


```
my_dictionary = {
#      KEY                    VALUE
#    vvvvvvv                 vvvvvvv
     "hello"               : "world",
     "squareOfTwo"         : 4,
     "theMeaningOfLife"    : 42,
     0                     : 1
}

my_dictionary["theMeaningOfLife"] = "wat"
wat = my_dictionary["theMeaningOfLife"]
print wat
```

Dictionaries

What if we wanted to get all the keys of a dictionary?

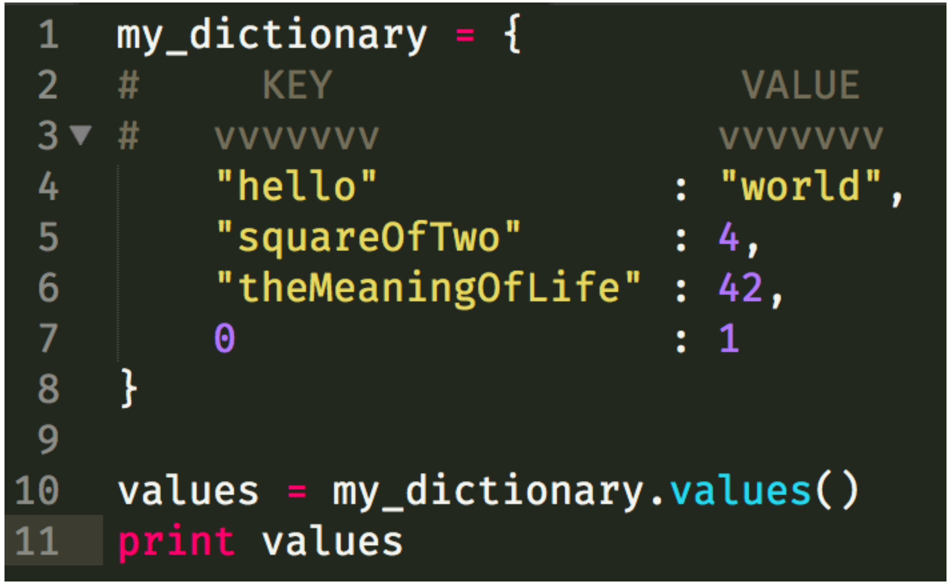


```
my_dictionary = {
#      KEY                          VALUE
#    vvvvvvv                       vvvvvvv
     "hello"                     : "world",
     "squareOfTwo"               : 4,
     "theMeaningOfLife"          : 42,
     0                           : 1
}

keys = my_dictionary.keys()
print keys
```

Dictionaries

What if we wanted to get all the values of a dictionary?

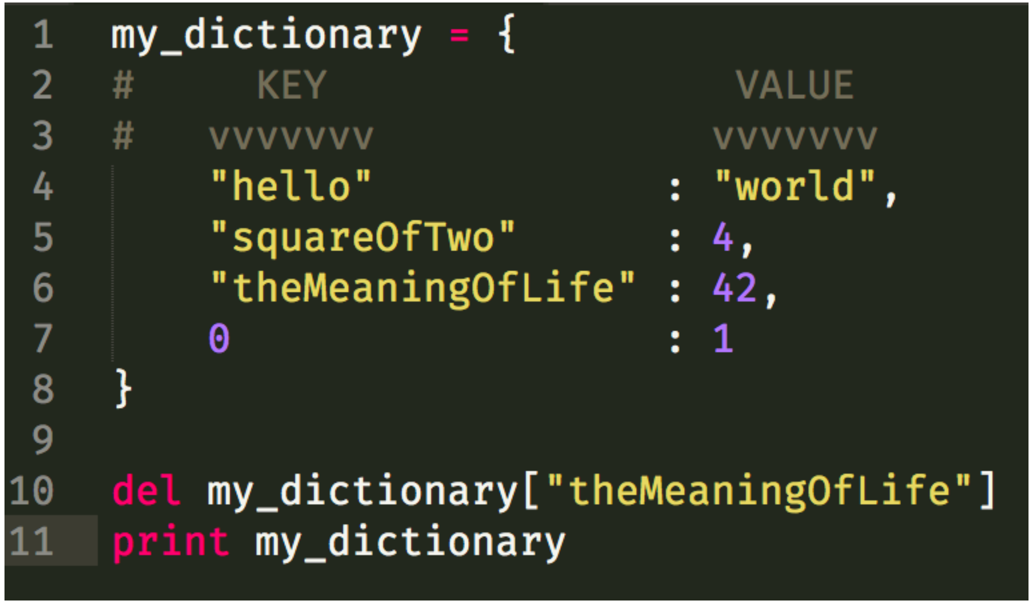


```
my_dictionary = {
#       KEY                           VALUE
#     vvvvvvv                        vvvvvvv
      "hello"                      : "world",
      "squareOfTwo"                : 4,
      "theMeaningOfLife"           : 42,
      0                            : 1
}

values = my_dictionary.values()
print values
```

Dictionaries

You can delete items too.

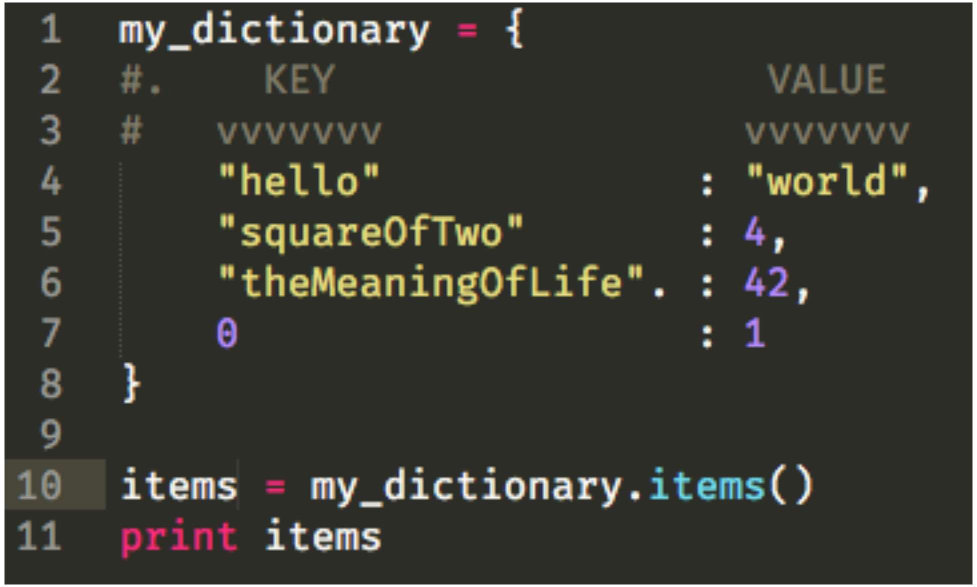


```python
my_dictionary = {
#       KEY                       VALUE
#    vvvvvvv                     vvvvvvv
     "hello"                   : "world",
     "squareOfTwo"             : 4,
     "theMeaningOfLife"        : 42,
     0                         : 1
}

del my_dictionary["theMeaningOfLife"]
print my_dictionary
```

Dictionaries

Dictionaries contain *entries.* An entry is a tuple containing the key and the value for every pair in the dictionary. You can get all entries like this:

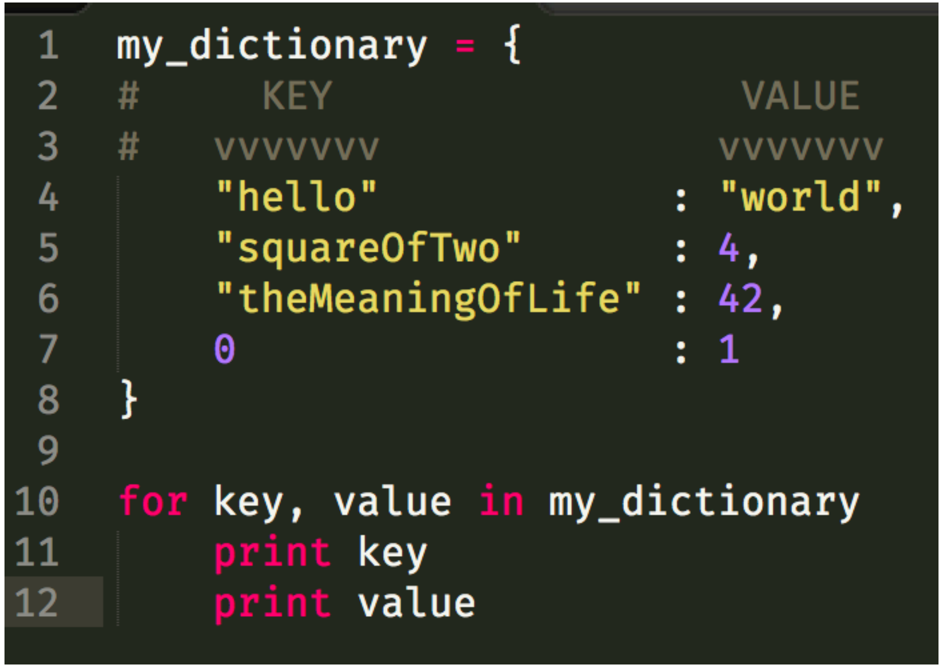


```
my_dictionary = {
#.      KEY                           VALUE
#     vvvvvvv                        vvvvvvv
      "hello"                      : "world",
      "squareOfTwo"                : 4,
      "theMeaningOfLife".          : 42,
      0                            : 1
}

items = my_dictionary.items()
print items
```

Dictionaries

One could, if one wanted, iterate over the entries in a dictionary. The syntax is a little interesting:

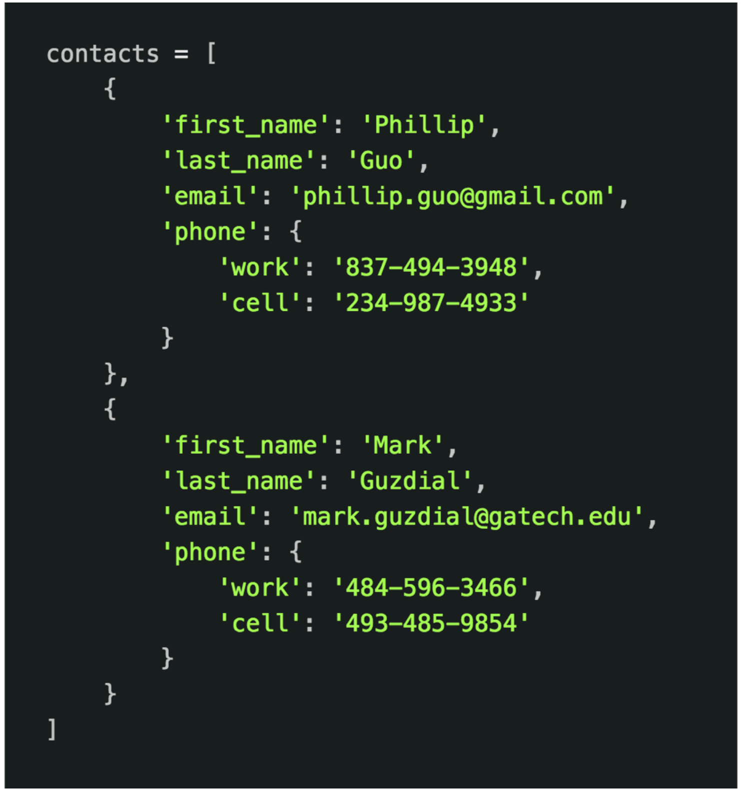


```
my_dictionary = {
#       KEY                      VALUE
#    vvvvvvv                    vvvvvvv
     "hello"                  : "world",
     "squareOfTwo"            : 4,
     "theMeaningOfLife"       : 42,
     0                        : 1
}

for key, value in my_dictionary
    print key
    print value
```

Dictionaries

Nesting can get pretty brutal, but it’s necessary. Dictionaries are a great way of storing data so you can get to it very quickly. And since they’re mutable, you can use them to keep information about things in your system efficiently.

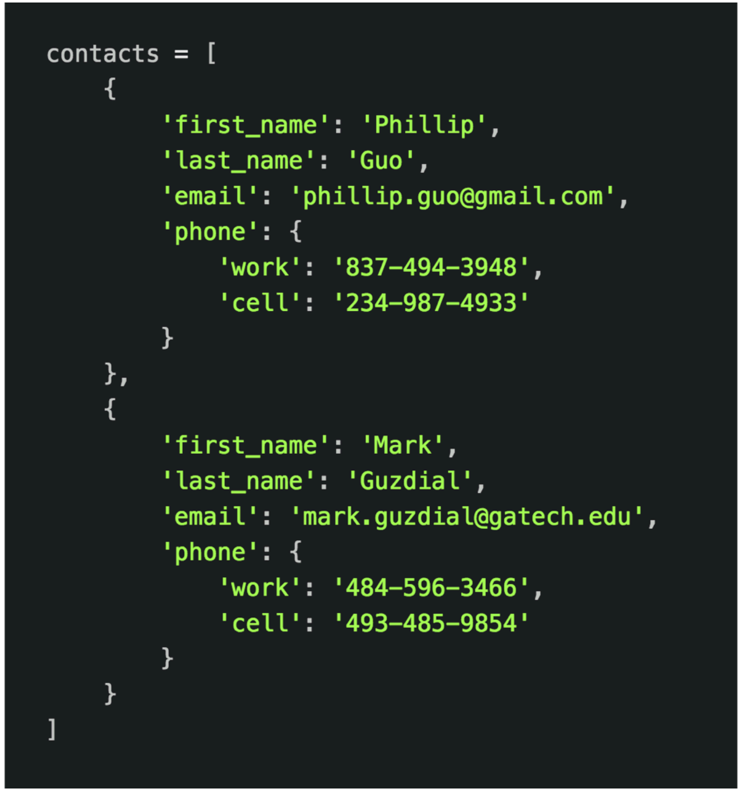


```
contacts = [
    {
        'first_name': 'Phillip',
        'last_name': 'Guo',
        'email': 'phillip.guo@gmail.com',
        'phone': {
            'work': '837-494-3948',
            'cell': '234-987-4933'
        }
    },
    {
        'first_name': 'Mark',
        'last_name': 'Guzdial',
        'email': 'mark.guzdial@gatech.edu',
        'phone': {
            'work': '484-596-3466',
            'cell': '493-485-9854'
        }
    }
]
```

Dictionaries

How do I get Phillip’s email address?

```
contacts[0][“email”]
```

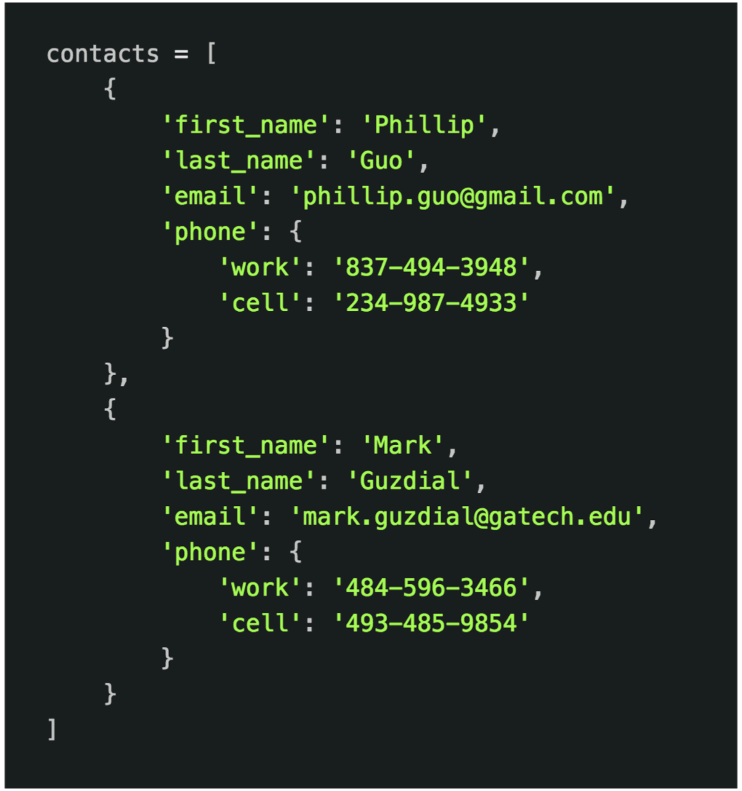


```
contacts = [
    {
        'first_name': 'Phillip',
        'last_name': 'Guo',
        'email': 'phillip.guo@gmail.com',
        'phone': {
            'work': '837-494-3948',
            'cell': '234-987-4933'
        }
    },
    {
        'first_name': 'Mark',
        'last_name': 'Guzdial',
        'email': 'mark.guzdial@gatech.edu',
        'phone': {
            'work': '484-596-3466',
            'cell': '493-485-9854'
        }
    }
]
```

Dictionaries

And Mark’s cell number?

```
contacts[1][“phone”][“cell”]
```



```
contacts = [
    {
        'first_name': 'Phillip',
        'last_name': 'Guo',
        'email': 'phillip.guo@gmail.com',
        'phone': {
            'work': '837-494-3948',
            'cell': '234-987-4933'
        }
    },
    {
        'first_name': 'Mark',
        'last_name': 'Guzdial',
        'email': 'mark.guzdial@gatech.edu',
        'phone': {
            'work': '484-596-3466',
            'cell': '493-485-9854'
        }
    }
]
```