Python programming for beginners

Stefan Zhauryd
Instructor

Module 4

Functions, Tuples, Dictionaries, and Data Processing

In this module, you will learn about:

- code structuring and the concept of function;
- function invocation and returning a result from a function;
- name scopes and variable shadowing;
- **tuples and their purpose, constructing and using tuples;**
- **dictionaries and their purpose, constructing and using dictionaries.**

Sequence types and mutability

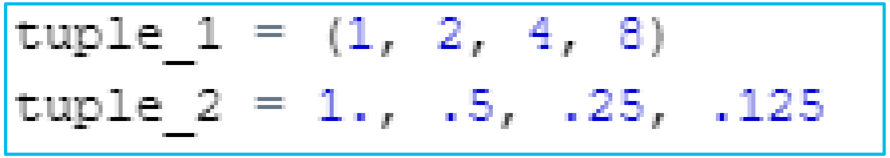
- [7, 7]
- []
- print("", 7, "")
- ()

- li = [7, 7.8, 9]

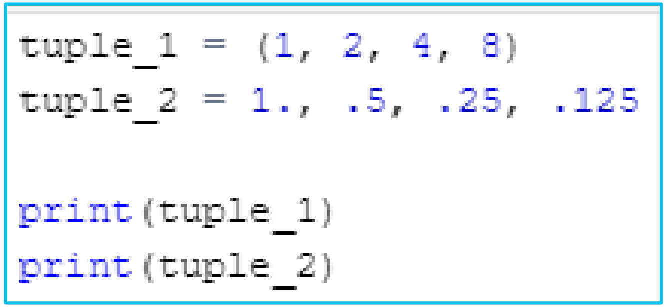
- li.append(8)
- del li[0]

What is a tuple?
X =

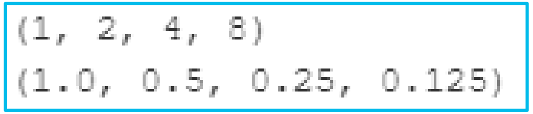
- Note: each tuple element may be of a different type (floating-point, integer, or any other not-as-yet-introduced kind of data).



```
tuple_1 = (1, 2, 4, 8)
tuple_2 = 1., .5, .25, .125
```

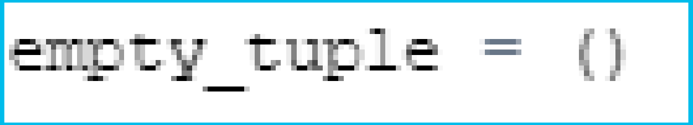


```
tuple_1 = (1, 2, 4, 8)
tuple_2 = 1., .5, .25, .125


print(tuple_1)
print(tuple_2)
```

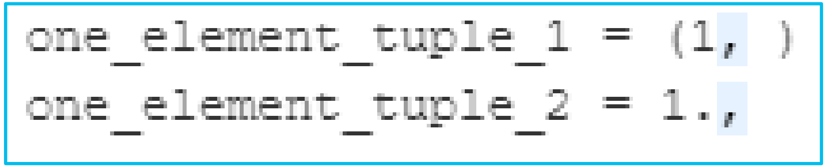


```
(1, 2, 4, 8)
(1.0, 0.5, 0.25, 0.125)
```

What is a tuple?

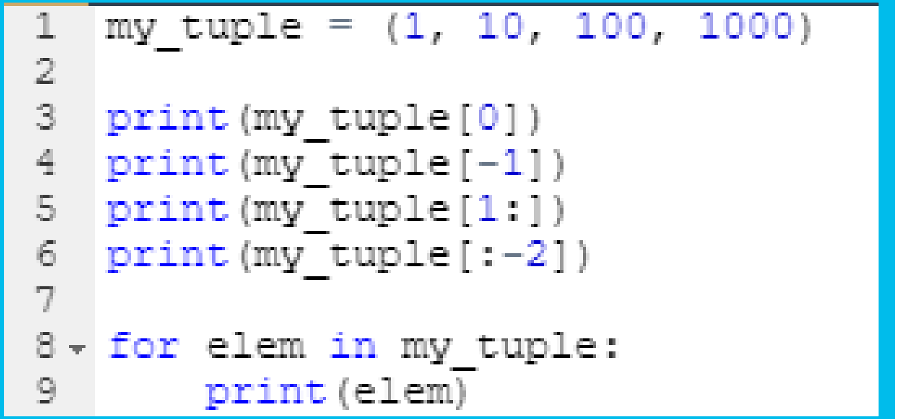


```
empty_tuple = ()
```

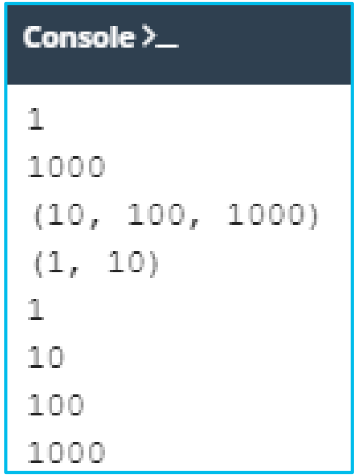


```
one_element_tuple_1 = (1, )
one_element_tuple_2 = 1.,
```

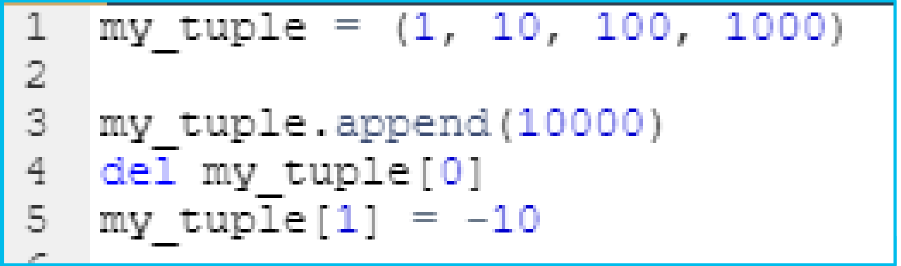
- The similarities may be misleading - don't try to modify a tuple's contents! It's not a list!



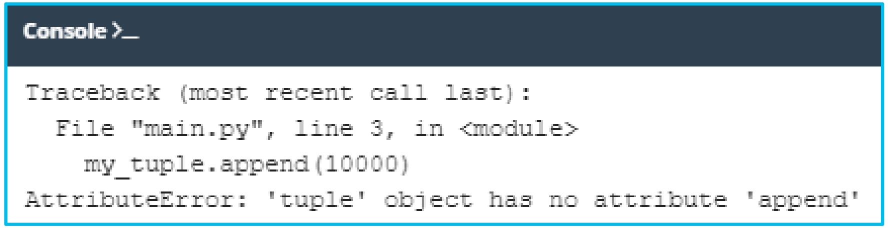
```
my_tuple = (1, 10, 100, 1000)

print(my_tuple[0])
print(my_tuple[-1])
print(my_tuple[1:])
print(my_tuple[:-2])

for elem in my_tuple:
    print(elem)
```



```
1
1000
(10, 100, 1000)
(1, 10)
1
10
100
1000
```

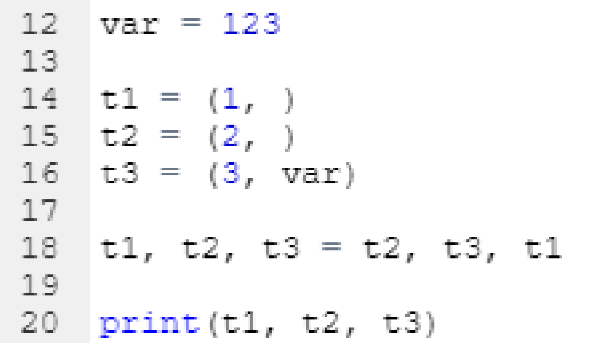


```
my_tuple = (1, 10, 100, 1000)

my_tuple.append(10000)
del my_tuple[0]
my_tuple[1] = -10
```

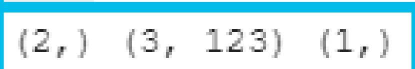


```
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    my_tuple.append(10000)
AttributeError: 'tuple' object has no attribute 'append'
```

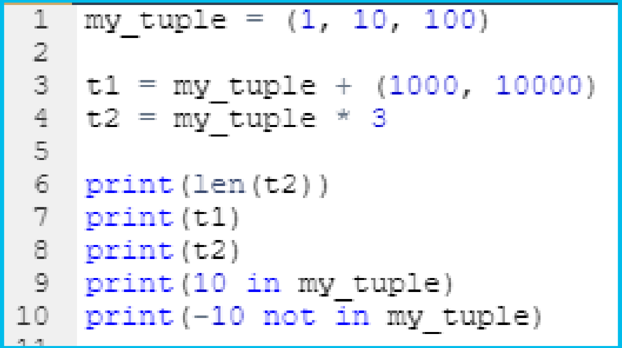
How to use a tuple: continued



```
var = 123

t1 = (1, )
t2 = (2, )
t3 = (3, var)

t1, t2, t3 = t2, t3, t1

print(t1, t2, t3)
```



```
(2,) (3, 123) (1,)
```

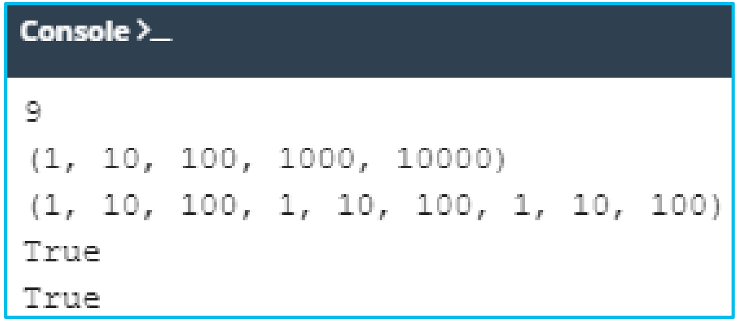


```
my_tuple = (1, 10, 100)

t1 = my_tuple + (1000, 10000)
t2 = my_tuple * 3

print(len(t2))
print(t1)
print(t2)
print(10 in my_tuple)
print(-10 not in my_tuple)
```

What else can tuples do for you?

- the **len()** function accepts tuples, and returns the number of elements contained inside;
- the **+** operator can join tuples together (we've shown you this already)
- the ***** operator can multiply tuples, just like lists;
- the **in** and **not in** operators work in the same way as in lists.

Note: the example presents one more important fact: a tuple's elements can be variables, not only literals. Moreover, they can be expressions if they're on the right side of the assignment operator.



```
9
(1, 10, 100, 1000, 10000)
(1, 10, 100, 1, 10, 100, 1, 10, 100)
True
True
```

What is a dictionary?

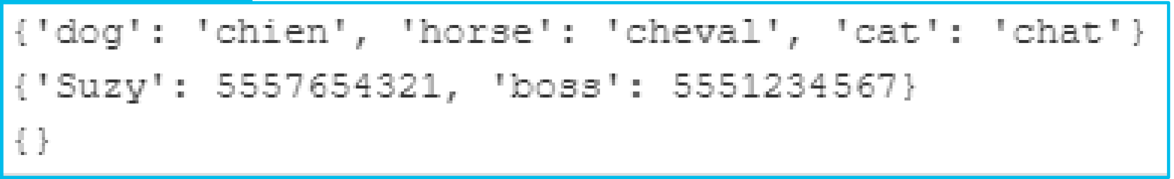
In Python's world, the word you look for is named a key. The word you get from the dictionary is called a value.

This means that a dictionary is a set of key-value pairs. Note:

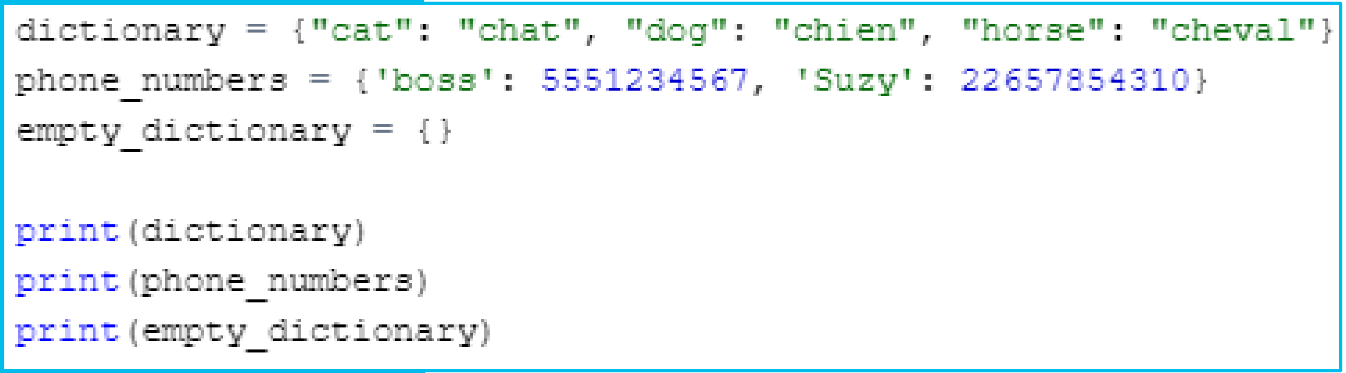
- each key must be **unique** - it's not possible to have more than one key of the same value;
- a key may be **any immutable type** of object: it can be **a number (integer or float), or even a string, but not a list**;
- a dictionary **is not a list** - a list contains a set of numbered values, while a dictionary holds pairs of values;
- the **len()** function works for dictionaries, too - it returns the numbers of key-value elements in the dictionary;
- a dictionary is a one-way tool - if you have an English-French dictionary, you can look for French equivalents of English terms, but not vice versa.

How to make a dictionary?

- In Python 3.6x dictionaries have become ordered collections by default. Your results may vary depending on what Python version you're using.

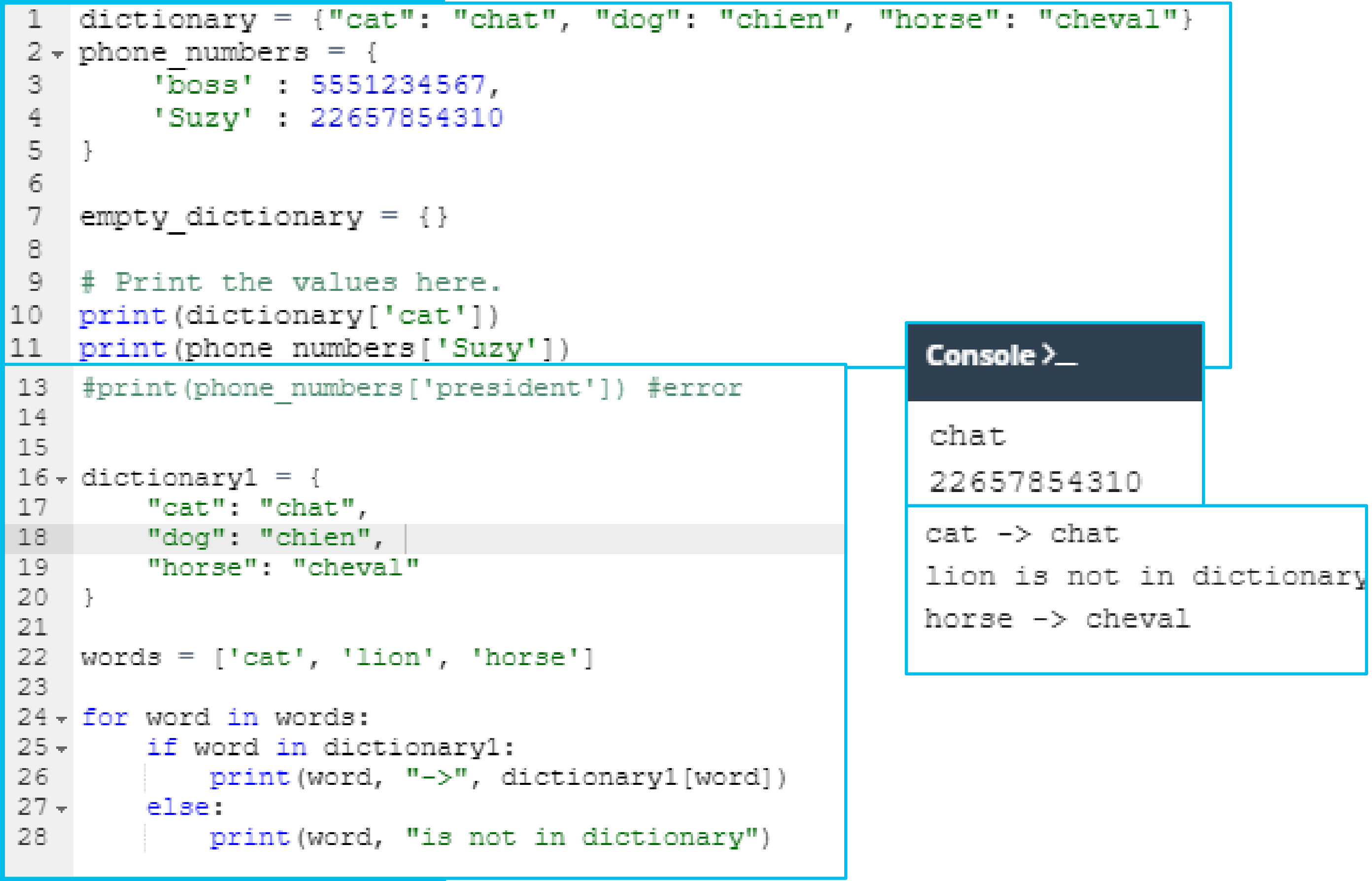


```
{'dog': 'chien', 'horse': 'cheval', 'cat': 'chat'}
{'Suzy': 5557654321, 'boss': 5551234567}
{}
```



```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}
phone_numbers = {'boss': 5551234567, 'Suzy': 22657854310}
empty_dictionary = {}

print(dictionary)
print(phone_numbers)
print(empty_dictionary)
```

Examples

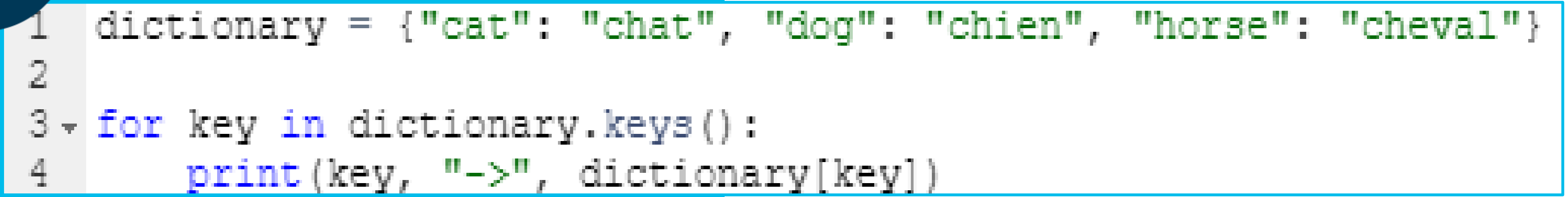


```python
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}
phone_numbers = {
    'boss' : 5551234567,
    'Suzy' : 22657854310
}

empty_dictionary = {}

# Print the values here.
print(dictionary['cat'])
print(phone_numbers['Suzy'])
```

```python
#print(phone_numbers['president']) #error


dictionary1 = {
    "cat": "chat",
    "dog": "chien",
    "horse": "cheval"
}

words = ['cat', 'lion', 'horse']

for word in words:
    if word in dictionary1:
        print(word, "->", dictionary1[word])
    else:
        print(word, "is not in dictionary")
```

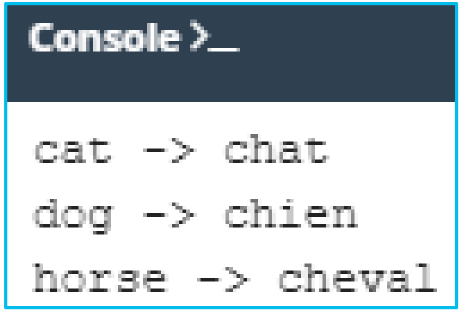
```
chat
22657854310
```

```
cat -> chat
lion is not in dictionary
horse -> cheval
```

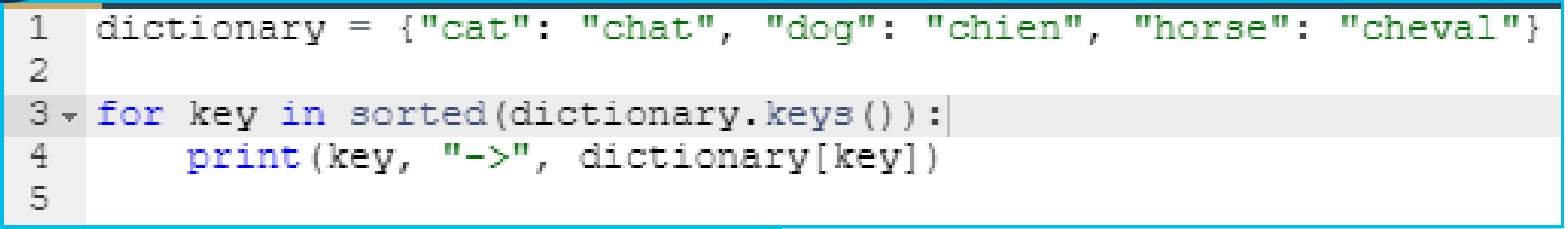


```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

for key in dictionary.keys():
    print(key, "->", dictionary[key])
```

How to use a dictionary: the **keys()**

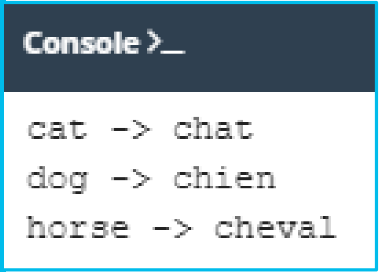


```
cat -> chat
dog -> chien
horse -> cheval
```



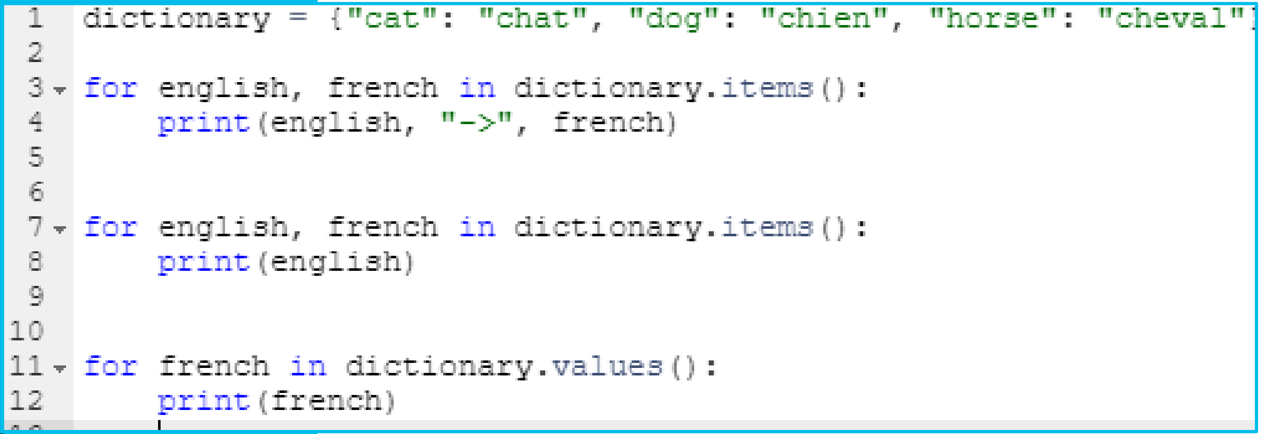
```python
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

for key in sorted(dictionary.keys()):
    print(key, "->", dictionary[key])

```

The **sorted()** function

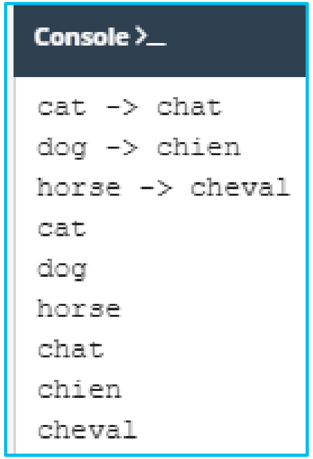


```
cat -> chat
dog -> chien
horse -> cheval
```

How to use a dictionary: The **items()** and **values()** methods



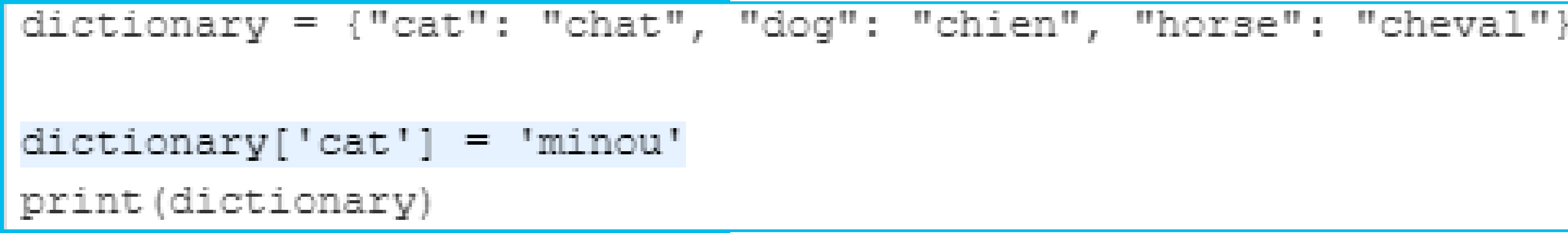
```python
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

for english, french in dictionary.items():
    print(english, "->", french)


for english, french in dictionary.items():
    print(english)


for french in dictionary.values():
    print(french)
```



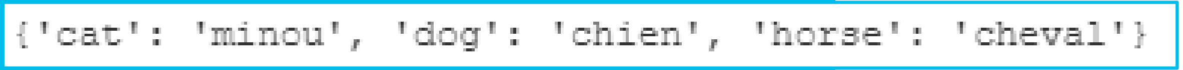
```
cat -> chat
dog -> chien
horse -> cheval
cat
dog
horse
chat
chien
cheval
```

How to use a dictionary: modifying and adding values

Assigning a new value to an existing key is simple - as dictionaries are fully mutable, there are no obstacles to modifying them.



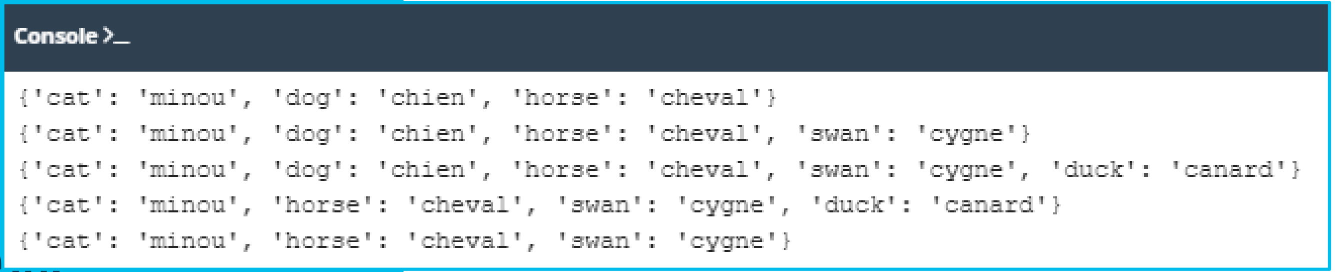
```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}


dictionary['cat'] = 'minou'
print(dictionary)
```



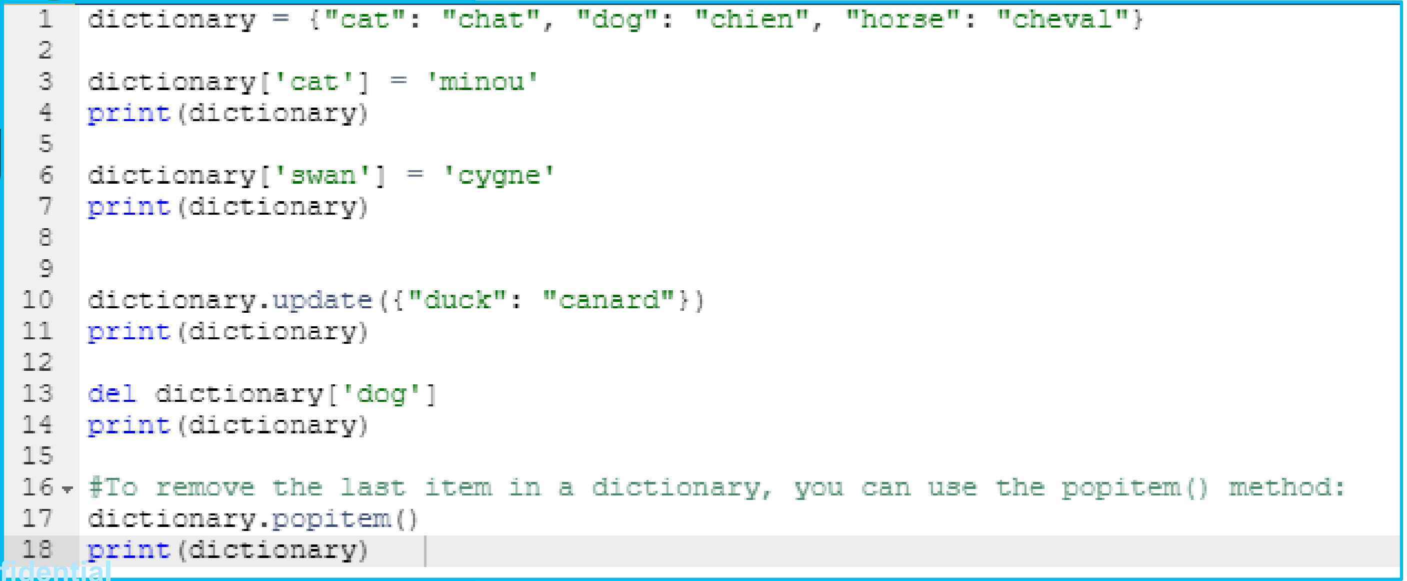
```
{'cat': 'minou', 'dog': 'chien', 'horse': 'cheval'}
```

Adding a new key
Removing

Note: removing a non-existing key causes an error.



```
{'cat': 'minou', 'dog': 'chien', 'horse': 'cheval'}
{'cat': 'minou', 'dog': 'chien', 'horse': 'cheval', 'swan': 'cygne'}
{'cat': 'minou', 'dog': 'chien', 'horse': 'cheval', 'swan': 'cygne', 'duck': 'canard'}
{'cat': 'minou', 'horse': 'cheval', 'swan': 'cygne', 'duck': 'canard'}
{'cat': 'minou', 'horse': 'cheval', 'swan': 'cygne'}
```



```python
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

dictionary['cat'] = 'minou'
print(dictionary)

dictionary['swan'] = 'cygne'
print(dictionary)


dictionary.update({"duck": "canard"})
print(dictionary)

del dictionary['dog']
print(dictionary)

#To remove the last item in a dictionary, you can use the popitem() method:
dictionary.popitem()
print(dictionary)
```

ЗАДАНИЯ

1) Прорешать всю классную работу
2) Выполнить все домашние задания

Почитать:

1) Byte of Python

Прочитать страницы -
стр. 88-94

ЗАДАНИЯ

Название файлов, которые вы отправляете мне в telegram:
Vasia_Pupkin_class_work_L6_P0.py

Формат сообщения которое вы присылаете мне
(после полного выполнения домашнего задания, только один раз) в Telegram:
Добрый день/вечер. Я Вася Пупкин, и это мои домашние задания к лекции 6 часть 0.
И отправляете файл

Крайний срок сдачи 07/10 в 21:00 (можно раньше, но не позже)

https://docs.github.com/articles/using-pull-requests

Q&A

Create your possibilities.

Bye bye.