Dictionaries!

Definition

Dictionary

A container data type that maps “keys” to their associated “values”.

Anatomy of a Dictionary

```
d = {'hansa': 4, 'kandula': 3, 'lumpy': 2, 'surus': 6}
```

Anatomy of a Dictionary

```
d = {'hansa': 4, 'kandula': 3, 'lumpy': 2, 'surus': 6}
```

This is a dictionary literal

Anatomy of a Dictionary

```
d = {'hansa': 4, 'kandula': 3, 'lumpy': 2, 'surus': 6}
```

This is a dictionary literal

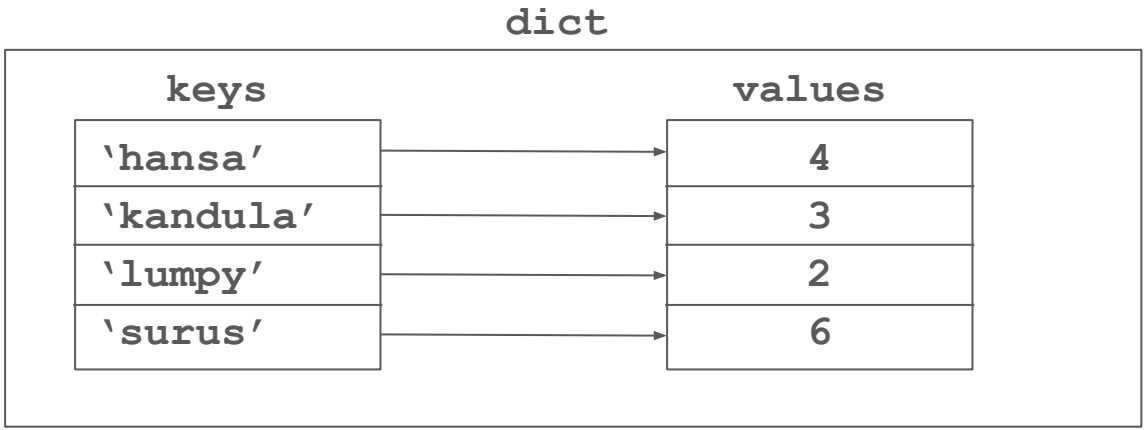
... but it's easier to visualize it this way:

Anatomy of a Dictionary

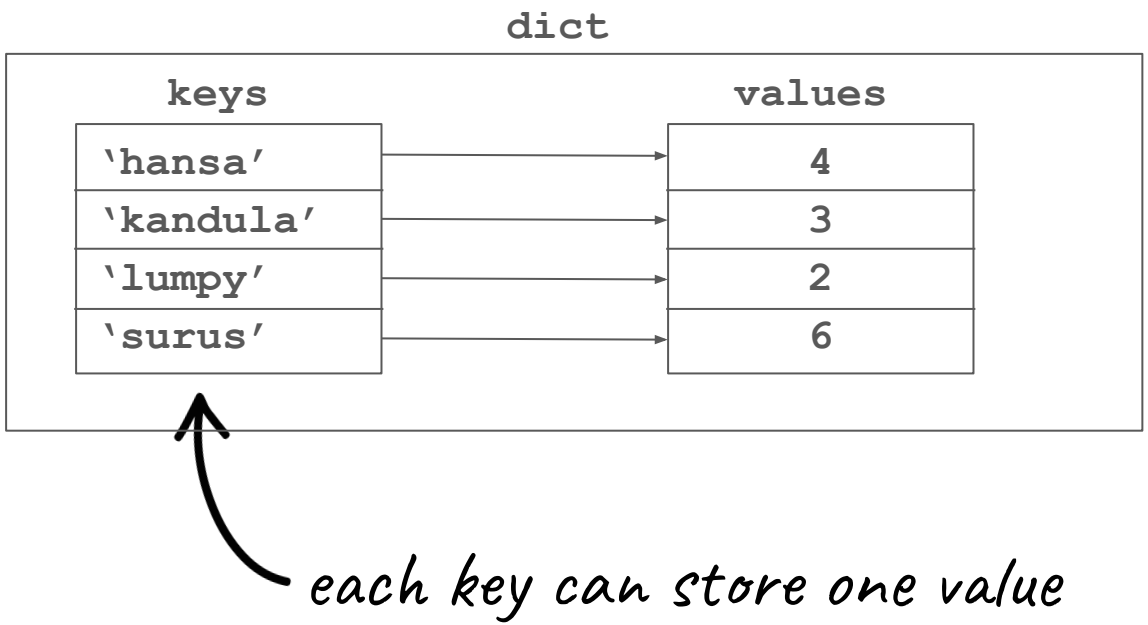
```
d = {'hansa': 4, 'kandula': 3, 'lumpy': 2, 'surus': 6}
```

This is a dictionary literal

... but it's easier to visualize it this way:

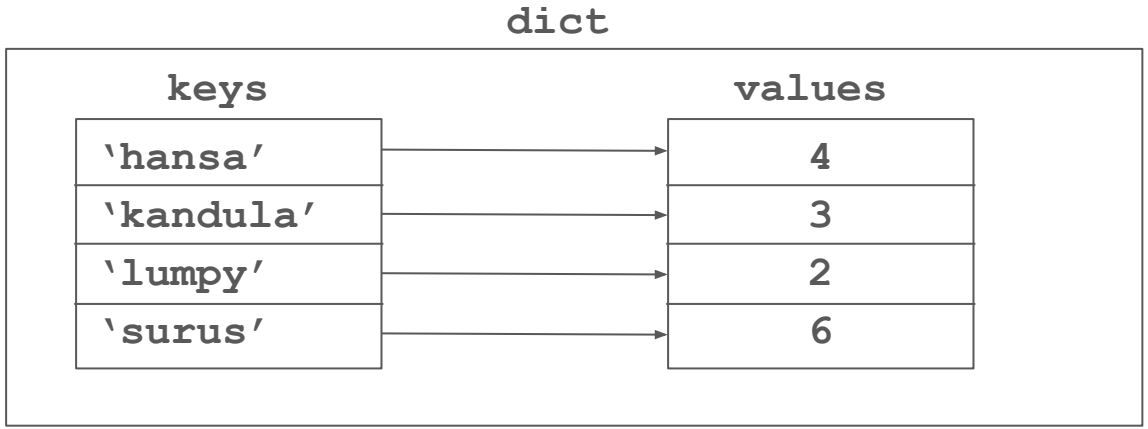


Anatomy of a Dictionary



Anatomy of a Dictionary - Get/Set

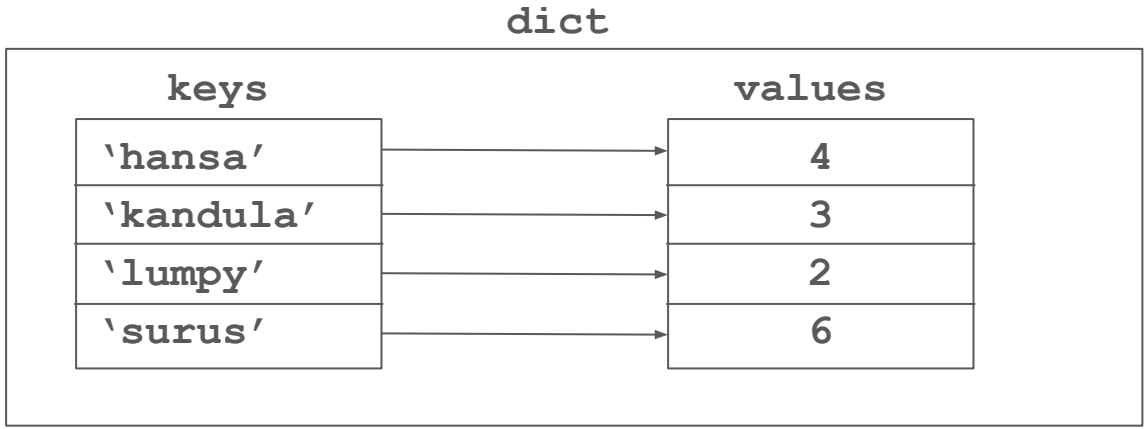
```
>>> d['hansa']
```



Anatomy of a Dictionary - Get/Set

```
>>> d['hansa']
```

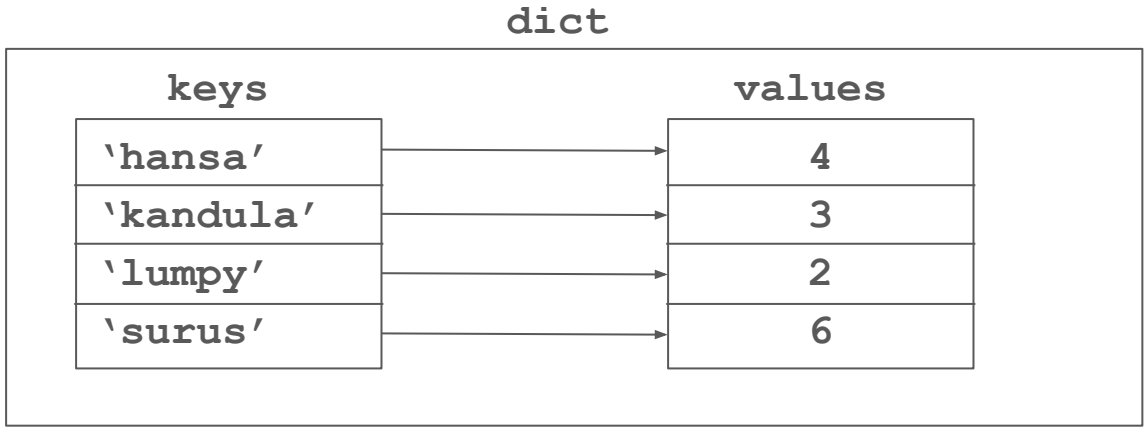
this operation is called "get"



Anatomy of a Dictionary - Get/Set

```
>>> d['hansa']

4
```

this operation is called "get"

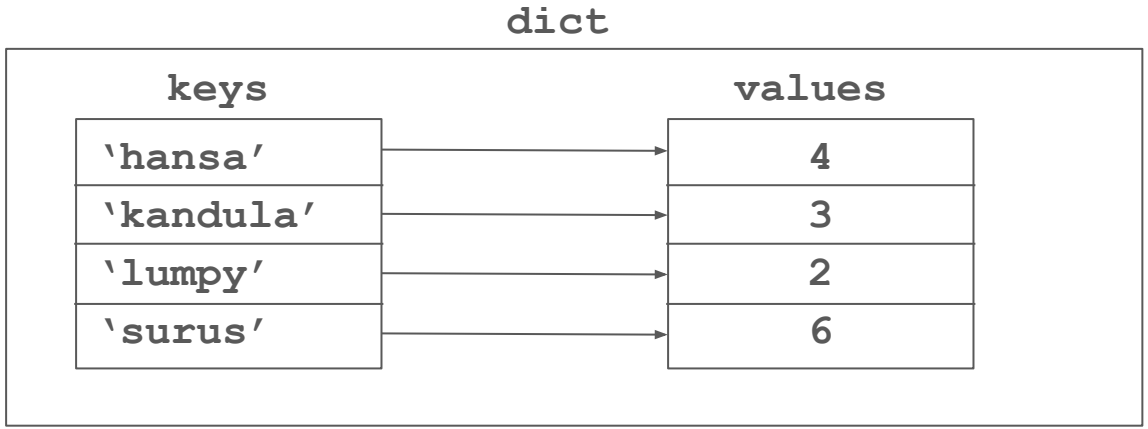


Anatomy of a Dictionary - Get/Set

```
>>> d['hansa']
```

```
4
```

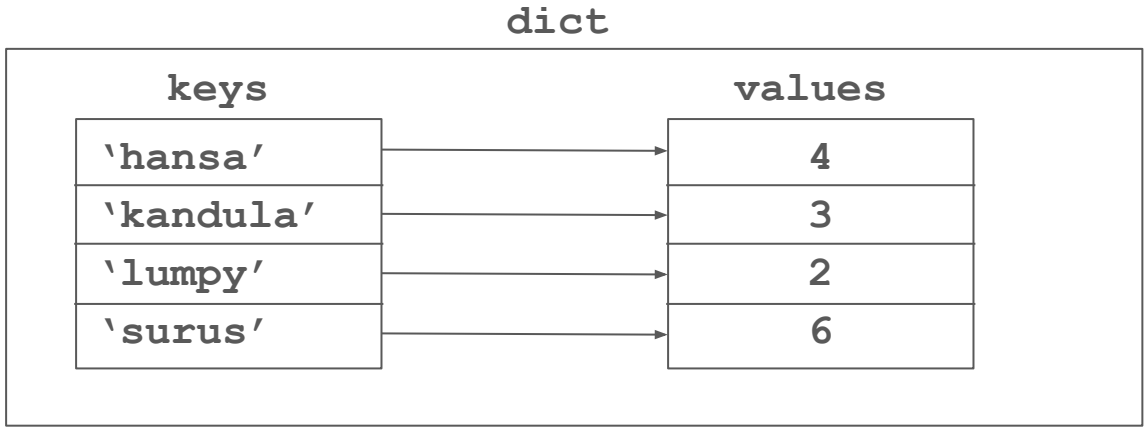
```
>>> d['hansa'] = 5
```



Anatomy of a Dictionary - Get/Set

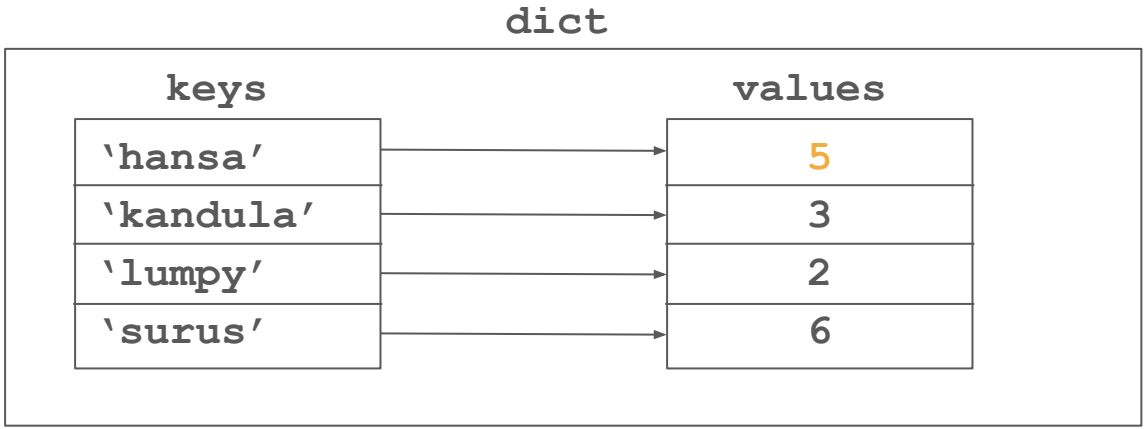
```
>>> d['hansa']
4

>>> d['hansa'] = 5
```

this operation is called "set"



Anatomy of a Dictionary - Get/Set

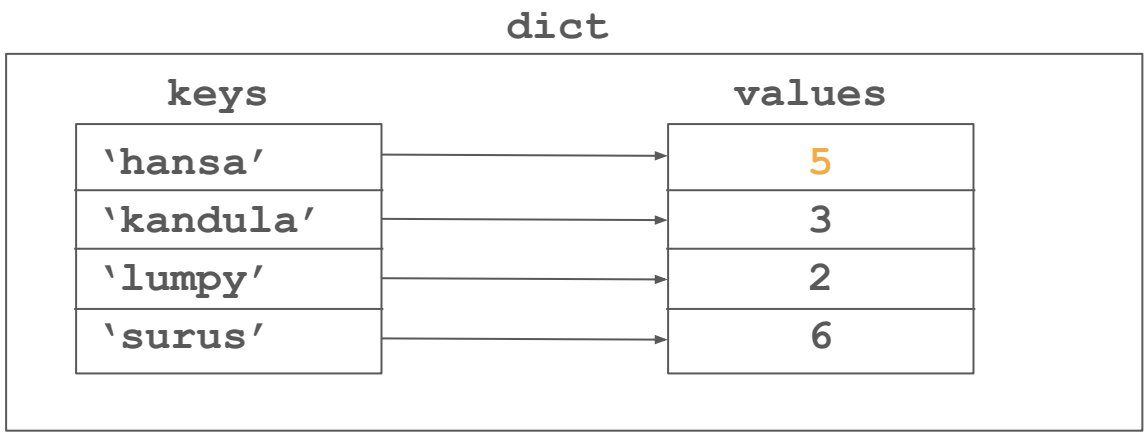
```
>>> d['hansa']

4

>>> d['hansa'] = 5
```



this operation is called "set"

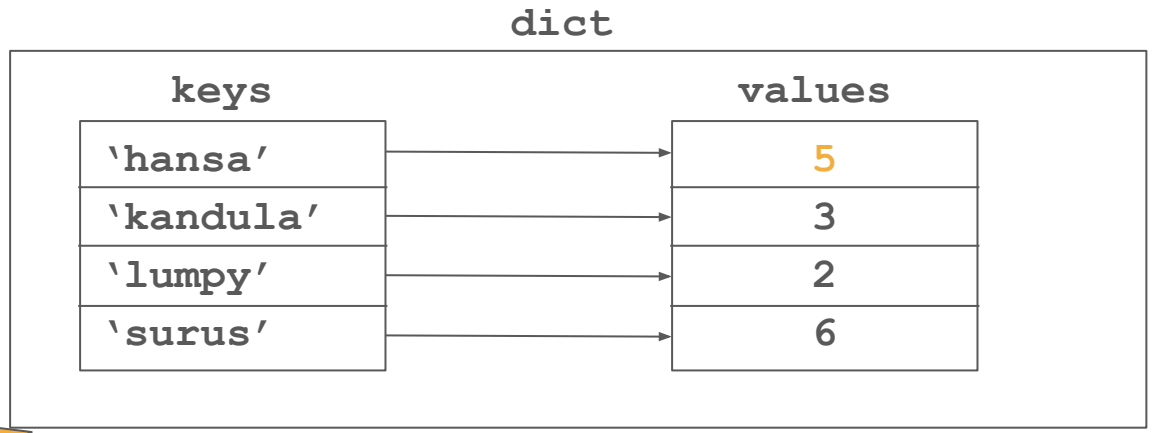
Anatomy of a Dictionary - Get/Set

```
>>> d['hansa']

4

>>> d['hansa'] = 5

>>> d['nick']
```



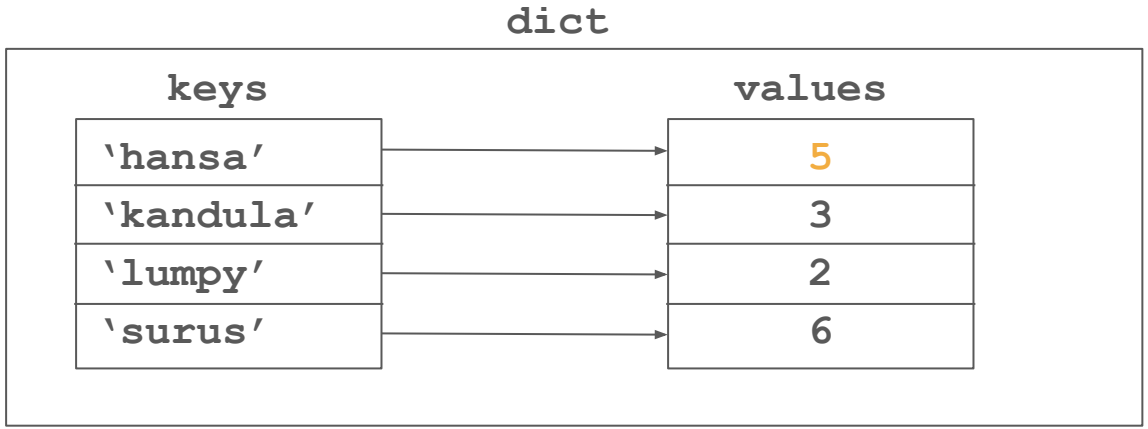
Anatomy of a Dictionary - Get/Set

```
>>> d['hansa']

4

>>> d['hansa'] = 5

>>> d['nil']
```



Anatomy of a Dictionary - Get/Set

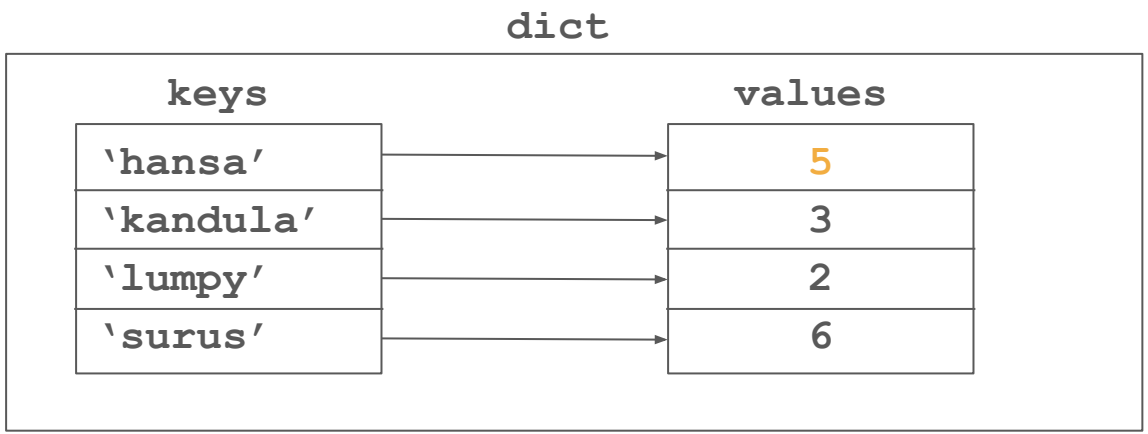
```
>>> d['hansa']
4
>>> d['hansa'] = 5
>>> d['nick']
```



"get" errors if the key is not in the dict.

Dictionaries + **in**

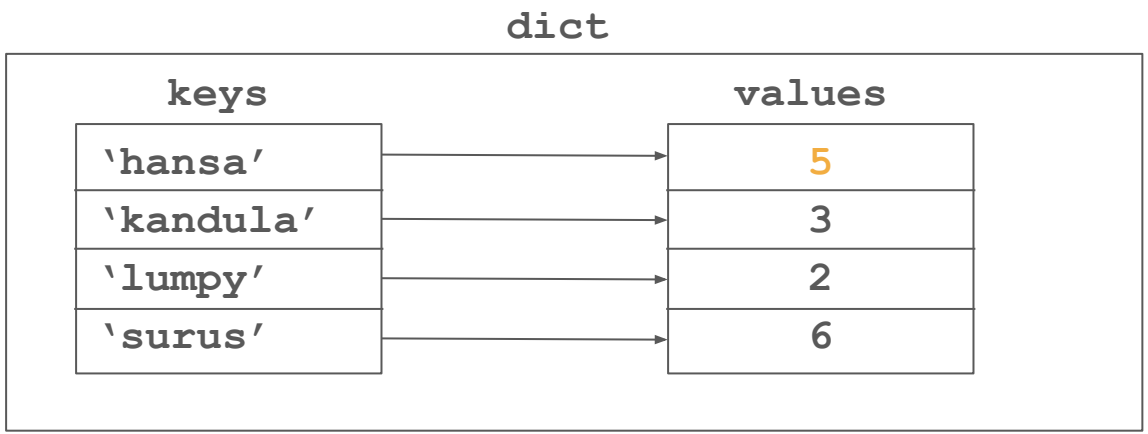
```
>>> 'hansa' in d
```



Dictionaries + **in**

```
>>> 'hansa' in d
```

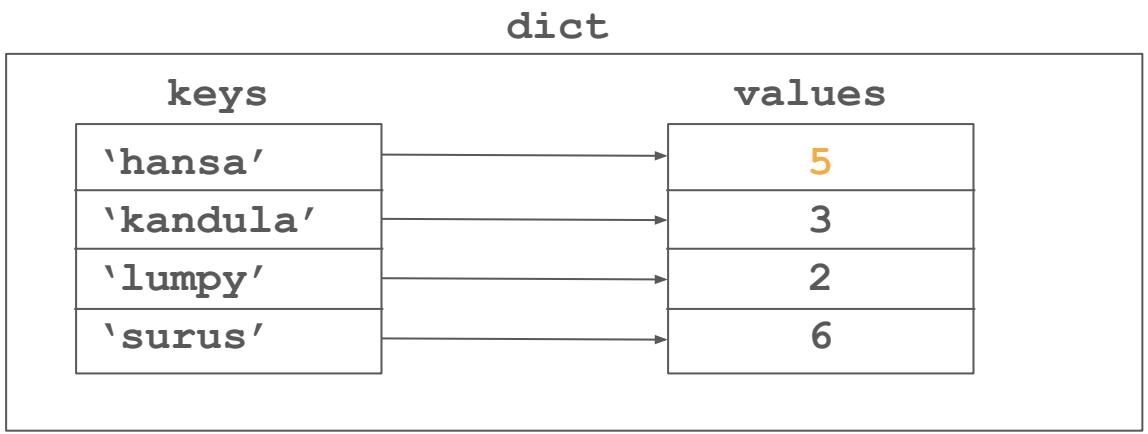
check if key is present



Dictionaries + **in**

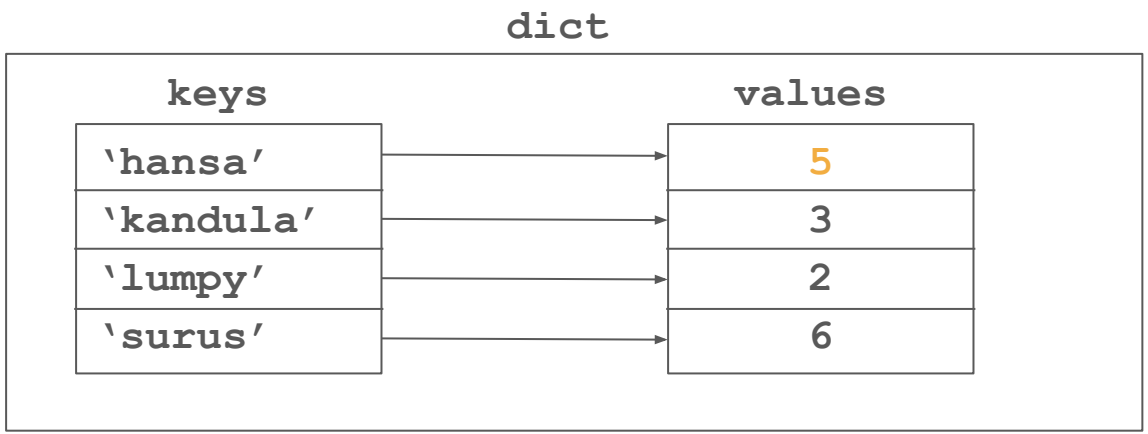
```
>>> 'hansa' in d
True
```

check if key is present



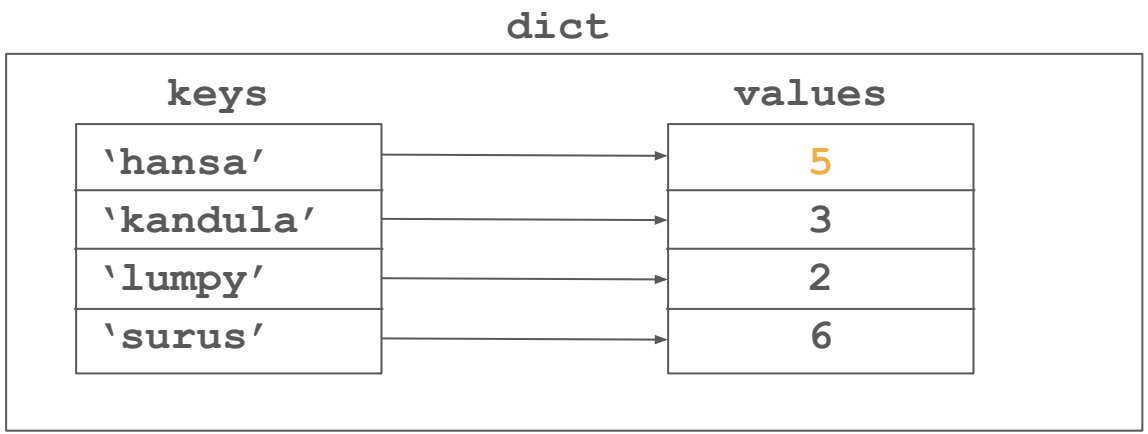
Dictionaries + **in**

```
>>> 'hansa' in d
True
>>> 'nick' not in d
```



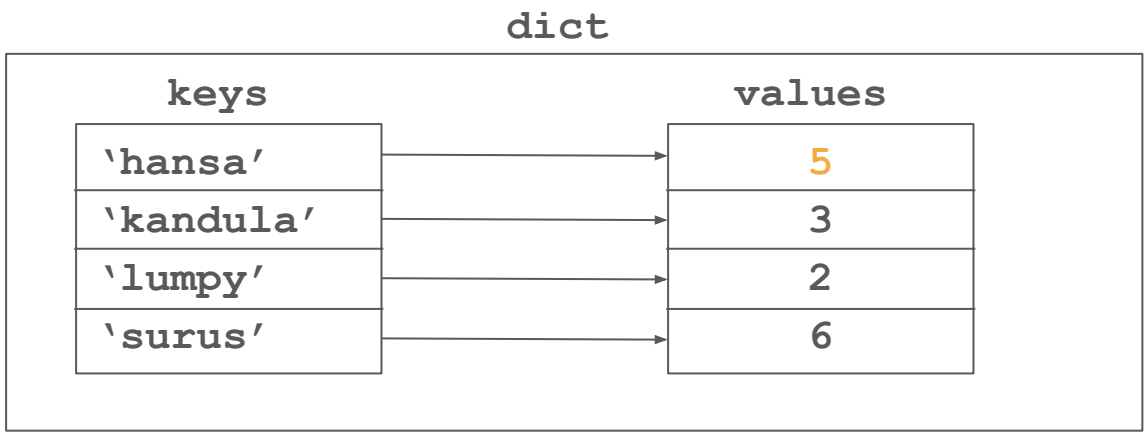
Dictionaries + **in**

```
>>> 'hansa' in d
True
>>> 'nick' not in d
True
```



Dictionaries + **in**

```
>>> 'hansa' in d
True
>>> 'nick' not in d
True
```



Common pattern:

Check if key is present. If it is, do something.

If it isn't, do something else.

Building a Dictionary

```
>>> d = {}
```

Building a Dictionary

```
>>> d = {}
```

create an empty dictionary

Building a Dictionary

```
>>> d = {}

>>> d['hansa'] = 3
```

Building a Dictionary

```
>>> d = {}

>>> d['hansa'] = 3
```

we can add keys using "set"!

Building a Dictionary

```
>>> d = {}

>>> d['hansa'] = 3

>>> d
```

we can add keys using "set"!

Building a Dictionary

```
>>> d = {}

>>> d['hansa'] = 3

>>> d

{'hansa': 3}
```

we can add keys using "set"!

Building a Dictionary

```
>>> d = {'hansa': 3}
```

Building a Dictionary

```
>>> d = {'hansa': 3}

>>> d['hansa'] += 2
```

Building a Dictionary

```
>>> d = {'hansa': 3}

>>> d['hansa'] += 2
```

we can get/set on the same line!
(same as d['hansa'] = d['hansa'] + 2)

Building a Dictionary

```
>>> d = {'hansa': 3}

>>> d['hansa'] += 2

>>> d
```

we can get/set on the same line!
(same as d['hansa'] = d['hansa'] + 2)

Building a Dictionary

```
>>> d = {'hansa': 3}

>>> d['hansa'] += 2

>>> d

{'hansa': 5}
```

we can get/set on the same line!
(same as d['hansa'] = d['hansa'] + 2)

Types of Dictionaries

- So far, we've seen dictionaries mapping from strings to ints

Types of Dictionaries

- So far, we've seen dictionaries mapping from strings to ints
  - This is not the only type of dictionary!

Types of Dictionaries

- So far, we've seen dictionaries mapping from strings to ints
  - This is not the only type of dictionary!
  - You can map from string/int/float to string/int/float...

Types of Dictionaries

- So far, we've seen dictionaries mapping from strings to ints
  - This is not the only type of dictionary!
  - You can map from string/int/float to string/int/float...
    - ...and more! (coming tomorrow)

Accessing a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
```

Accessing a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> d.keys()
```

Accessing a Dictionary’s Keys

```
>>> d = {‘Gates’: 23, ‘MemChu’: 116, ‘Tresidder’: 57}

>>> d.keys()

dict_keys(['Gates', 'MemChu', 'Tresidder'])
```

Accessing a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> d.keys()

dict_keys(['Gates', 'MemChu', 'Tresidder'])
```

iterable collection of all the keys.

iterable means it can be used in foreach

Accessing a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> list(d.keys())
```

Accessing a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> list(d.keys())

['Gates', 'MemChu', 'Tresidder']
```

Accessing a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> list(d.keys())
['Gates', 'MemChu', 'Tresidder']
```

we are using list() to convert d.keys() into a list

Accessing a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
```

Accessing a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> list(d.values())
```

Accessing a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> list(d.values())
```

we are using list() to convert d.values() into a list

Accessing a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> list(d.values())
[23, 116, 57]
```

we are using list() to convert d.values() into a list

Looping over a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
```

Looping over a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> for building in d.keys():
```

Looping over a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> for building in d.keys():

...     print(building)
```

Looping over a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building in d.keys():
...     print(building)
Gates
MemChu
Tresidder
```

Looping over a Dictionary's Keys

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building in d.keys():
...     print(building)
Gates
MemChu
Tresidder
```

we can use foreach on the dictionary's keys!

Looping over a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
```

Looping over a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> for age in d.values():
```

Looping over a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> for age in d.values():

...     print(age)
```

Looping over a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for age in d.values():
...     print(age)
23
116
57
```

Looping over a Dictionary's Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for age in d.values():
...     print(age)
23
116
57
```

we can use foreach on the dictionary's values!

Looping over a Dictionary's Keys and Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
```

Looping over a Dictionary's Keys and Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> for building, age in d.items():
```

Looping over a Dictionary's Keys and Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}

>>> for building, age in d.items():
```

items() gives us key, value pairs

Looping over a Dictionary's Keys and Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building, age in d.items():
...     print(building, 'is', age, 'years old.')
```

items() gives us key, value pairs

Looping over a Dictionary's Keys and Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building, age in d.items():
...     print(building, 'is', age, 'years old.')
Gates is 23 years old.
MemChu is 116 years old.
Tresidder is 57 years old.
```

items() gives us key, value pairs

Looping over a Dictionary's Keys and Values

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building, age in d.items():
...     print(building, 'is', age, 'years old.')
Gates is 23 years old.
MemChu is 116 years old.
Tresidder is 57 years old.
```

print() will automatically concatenate args separated by commas!

Printing with sep=

```
>>> d = {‘Gates’: 23, ‘MemChu’: 116, ‘Tresidder’: 57}
>>> for building, age in d.items():
...     print(building, age, sep=‘: ’)
```

Printing with sep=

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building, age in d.items():
...     print(building, age, sep=': ')
```

sep is an optional argument like end!

Printing with sep=

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building, age in d.items():
...     print(building, age, sep=': ')
Gates: 23
MemChu: 116
Tresidder: 57
```

sep is an optional argument like end!

Printing with sep=

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building, age in d.items():
...     print(building, age, sep=': ')
Gates: 23
MemChu: 116
Tresidder: 57
```

the separating string will be printed between the arguments you pass into print()

Printing with sep=

```
>>> d = {'Gates': 23, 'MemChu': 116, 'Tresidder': 57}
>>> for building, age in d.items():
...     print(building, age, sep=': ')
Gates: 23
MemChu: 116
Tresidder: 57
```

the default is sep=' '
(insert space)

Dictionary in the wild 1

มีคำมาให้ใน list นับจำนวนครั้งของคำที่อยู่ใน list

[

‘English’, ‘Math’, ‘Math’, ‘Science’, ‘English’, ‘Science’, ‘PE’, ‘English’, ‘Thai’, ‘Science’

]

Dictionary in the wild 2

จากข้อ 1 ต้องการรู้ว่ามีคำไหนบ้างที่อยู่ใน **list 1** ครั้ง, **2** ครั้ง, **...**

Dictionary in the wild 3

ไฟล์เก็บข้อมูล email address และ password ของผู้ใช้งาน แต่ละบรรทัดอยู่ในรูปแบบ

email [วรรค] password

ต้องการเขียนโปรแกรมเพื่อเช็คว่า email address และ password ที่กรอกเข้ามาถูกต้องหรือเปล่า (สมมุติว่าโปรแกรมรันอยู่ตลอดเวลา สามารถเช็คได้เรื่อยๆ)