CS Bridge, Lecture 16

Dictionaries

Today’s questions

How can I organize my data so it’s easier to use?

How can I organize my data so it’s easier to use?

Think/Share:

Store names of habitat animals and their corresponding diet

elephant bear otter platypus

clams grass shrimp berries

Task - Relating data with each other

```
['elephant', ‘bear', ‘otter', ‘platypus']
['grass', ‘berries', ‘clams’, ‘shrimp']
```

Task - Relating data with each other

```
['elephant', ‘bear', ‘otter', ‘platypus']
['grass', ‘berries', ‘clams’, ‘shrimp']
```

These pieces of information are linked!

Task - Relating data with each other

```
['elephant', ‘bear', ‘otter', ‘platypus']
['grass', ‘berries', ‘clams’, ‘shrimp']
```

These pieces of information are linked!

Can we store them so they're associated with each other?

Dictionaries!

Definition

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

Anatomy of a Dictionary

```
name_of_dic = {}

name_of_dic = {'elephant': 'grass', 'bear': ‘berries',
'otter': ‘clams’, 'platypus': ‘shrimp'}
```

Anatomy of a Dictionary

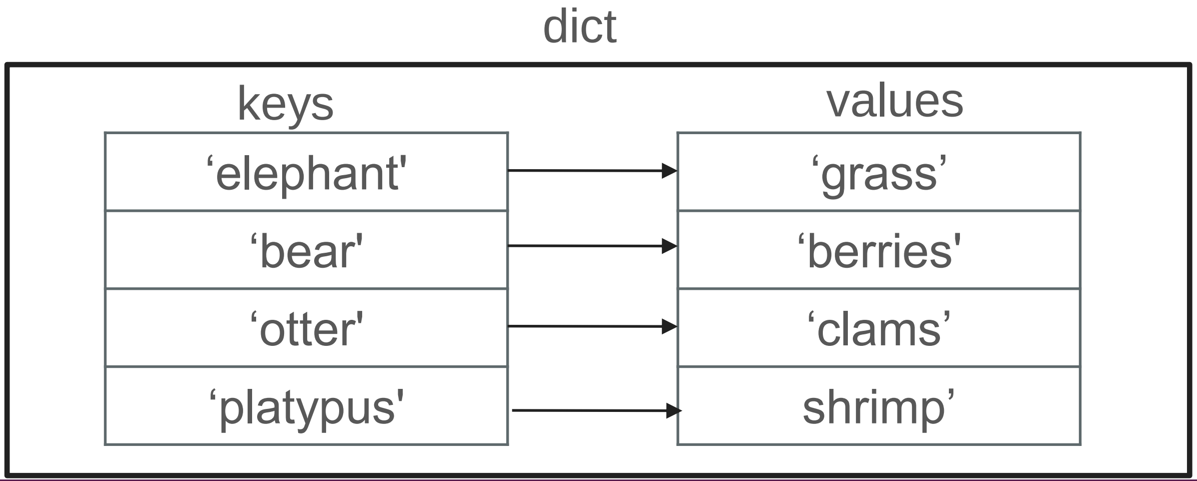
```
name_of_dic = {'elephant': 'grass', 'bear': ‘berries',
'otter': ‘clams’, 'platypus': ‘shrimp'}
```

This is a dictionary literal

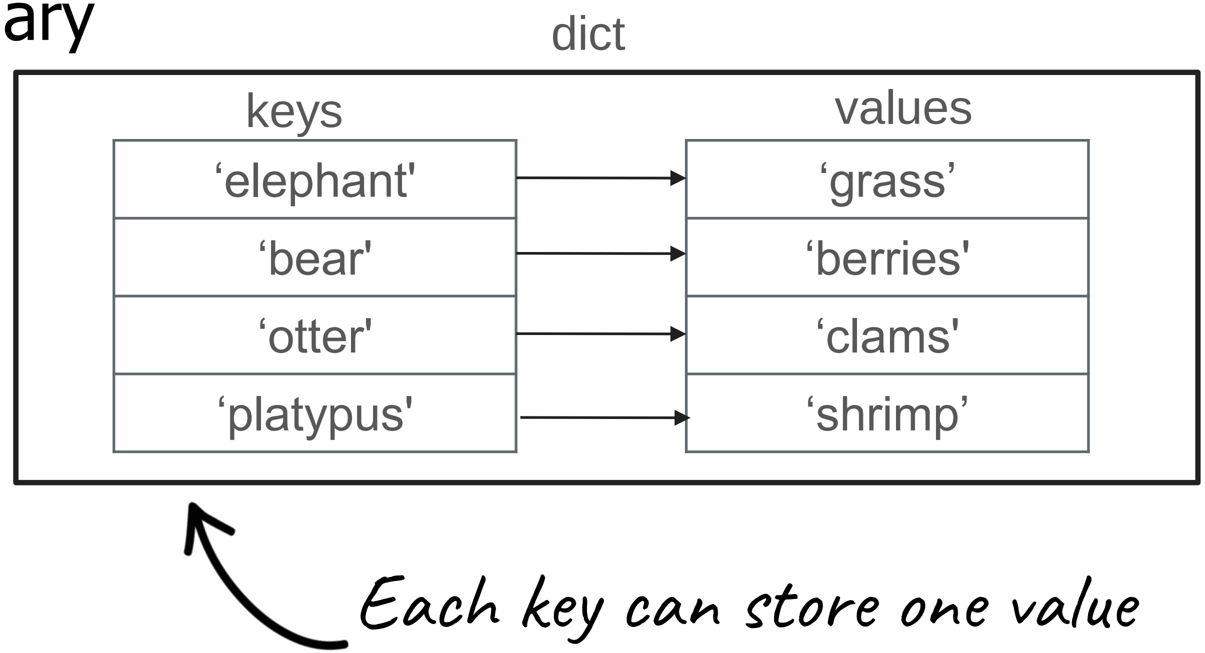
Anatomy of a Dictionary

```
name_of_dic = {'elephant': 'grass', 'bear': `berries',
'otter': `clams', 'platypus': `shrimp'}
```

It is easier to visualize it this way:

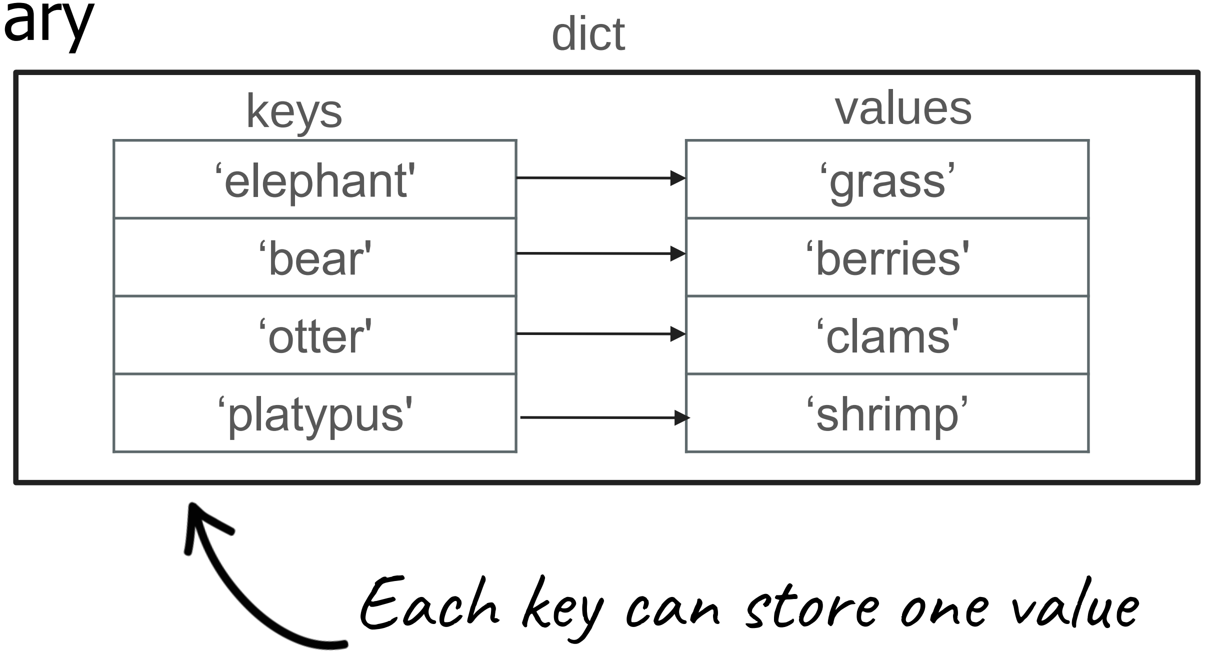


Anatomy of a Dictionary



Anatomy of a Dictionary

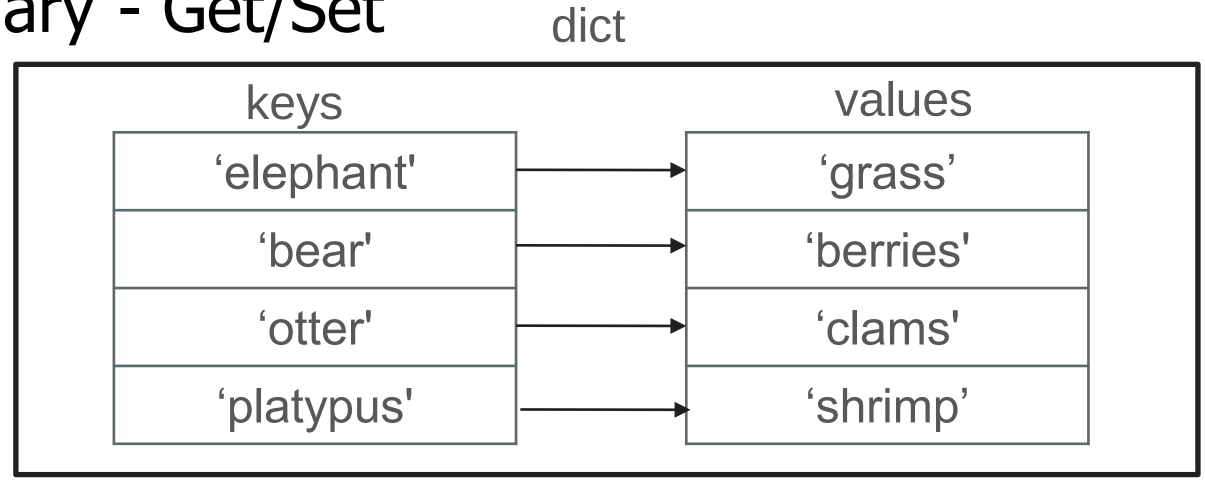
```
>>> d[‘elephant’]
```



Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
```

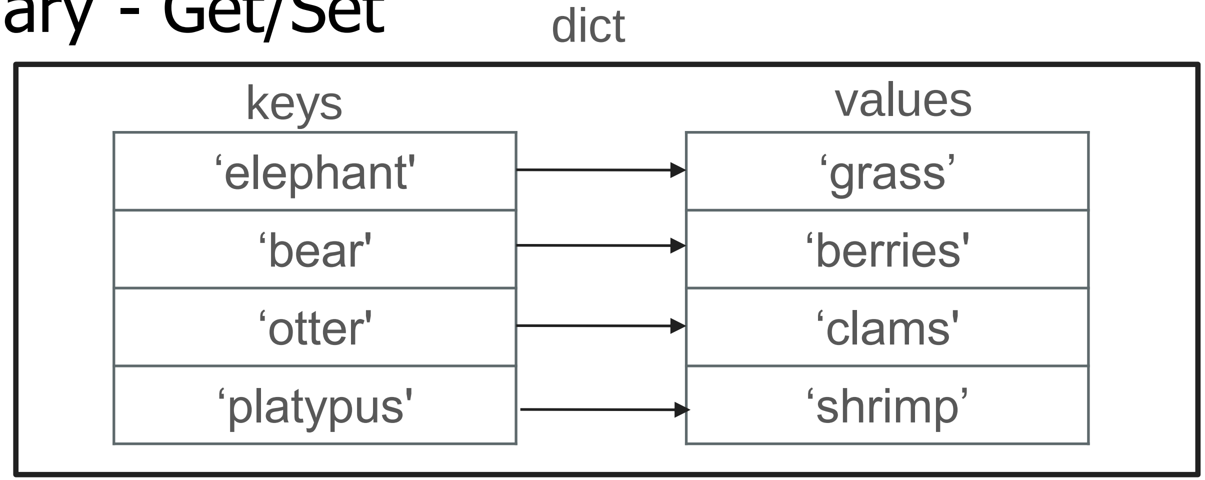
This operation is called "get"



Anatomy of a Dictionary - Get/Set

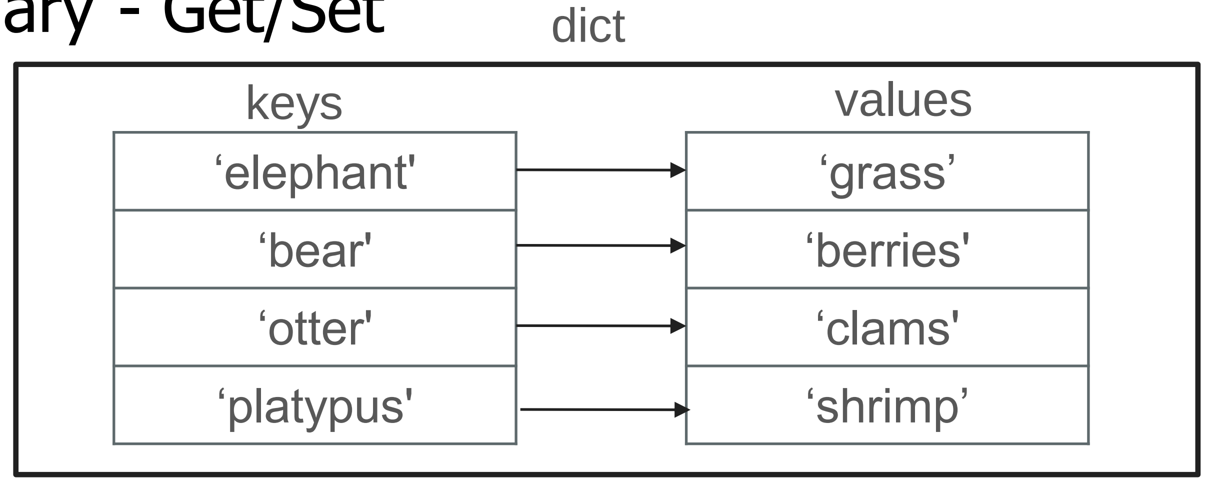
```
>>> d[‘elephant’]
```

This operation is called “get”



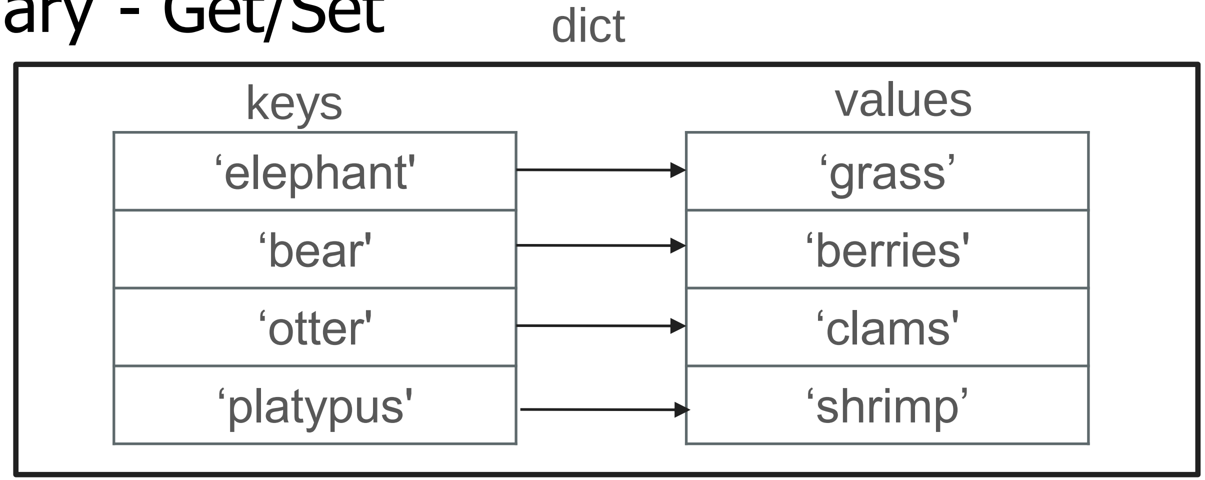
Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
'grass'
```



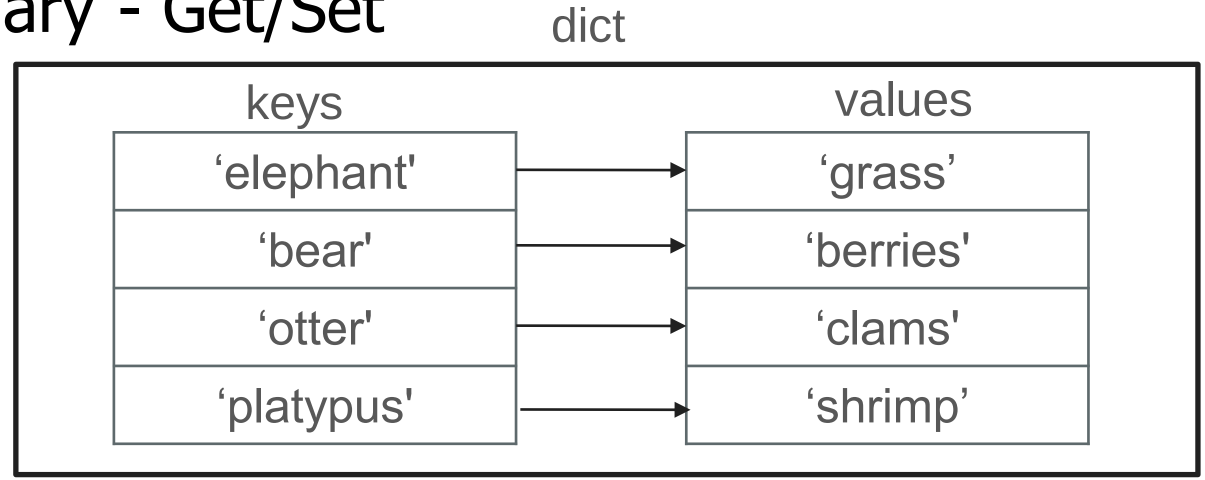
Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
'grass'
```



Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']
'grass'
```



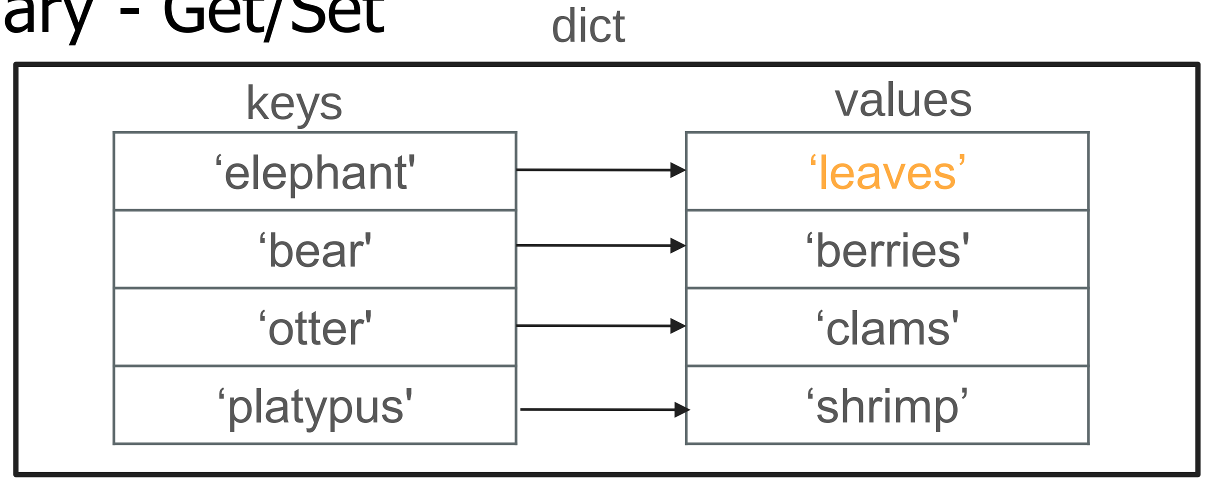
```
>>> d['elephant'] = 'leaves'
```

This operation is called "set"

Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']

'grass'
```



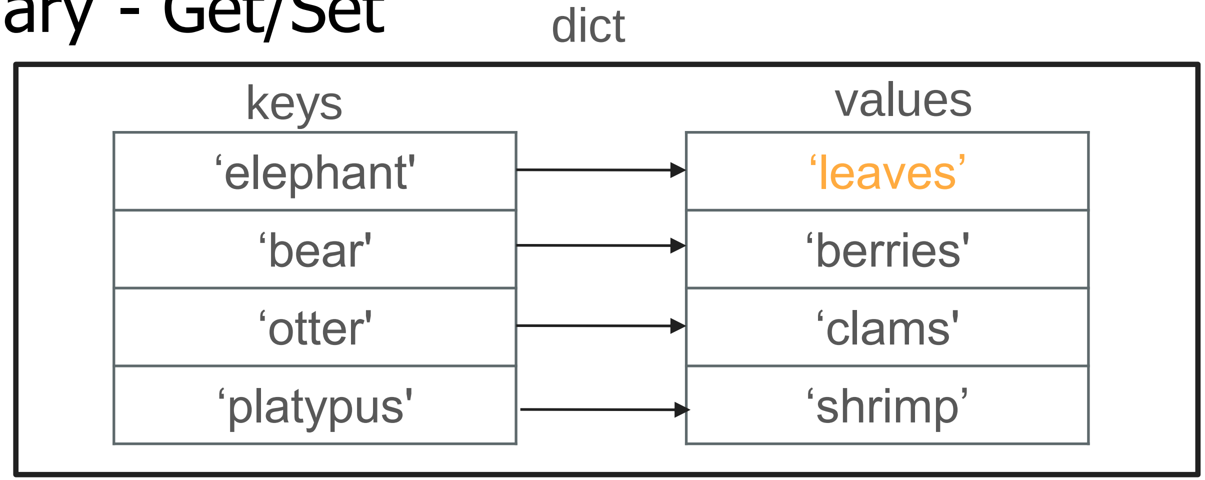
```
>>> d['elephant'] = 'leaves'
```

This operation is called "set"

Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']

'grass'
```



```
>>> d['elephant'] = 'leaves'
>>> d['cat']
```

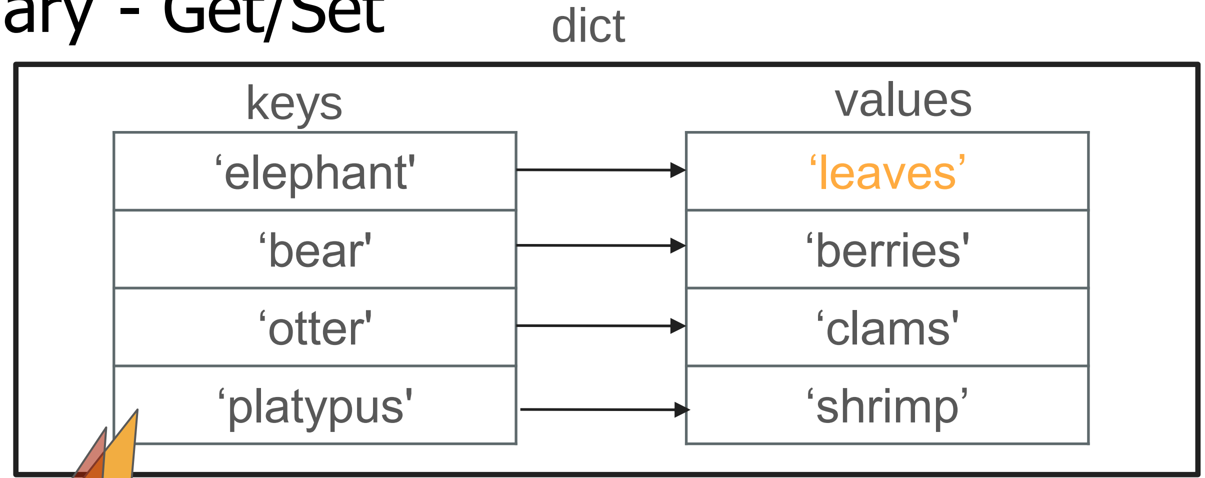
Anatomy of a Dictionary - Get/Set

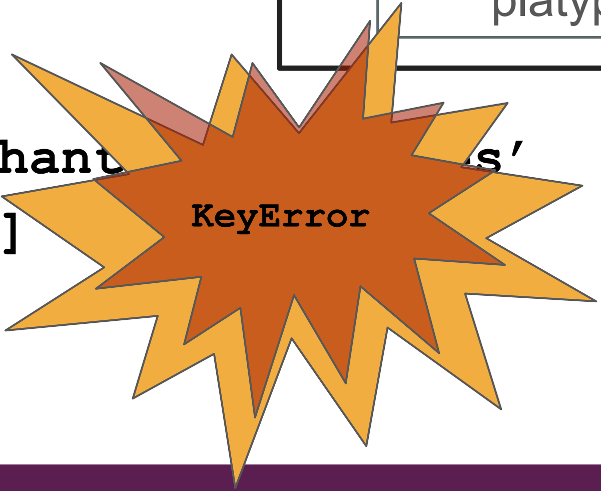
```
>>> d['elephant']

'grass'



>>> d['elephant                s'
>>> d['cat']
```

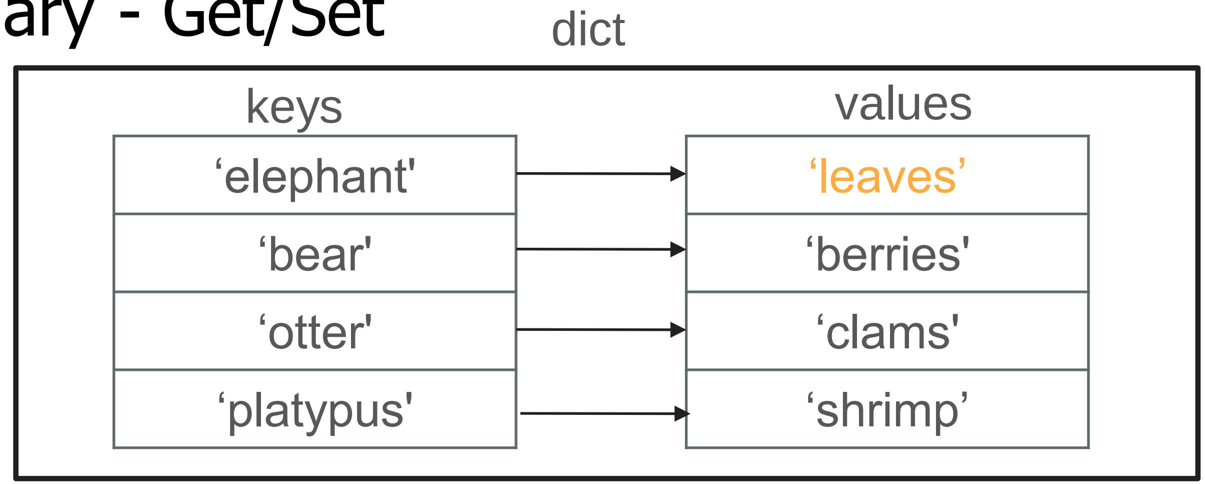




Anatomy of a Dictionary - Get/Set

```
>>> d['elephant']

'grass'
```

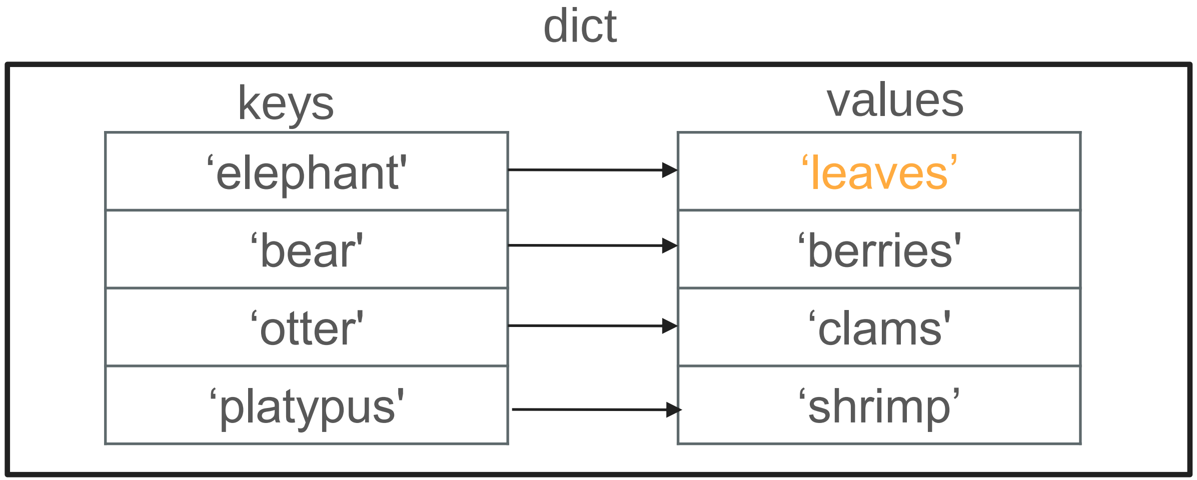


```
>>> d['elephant'] = 'leaves'
>>> d['cat']
```

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

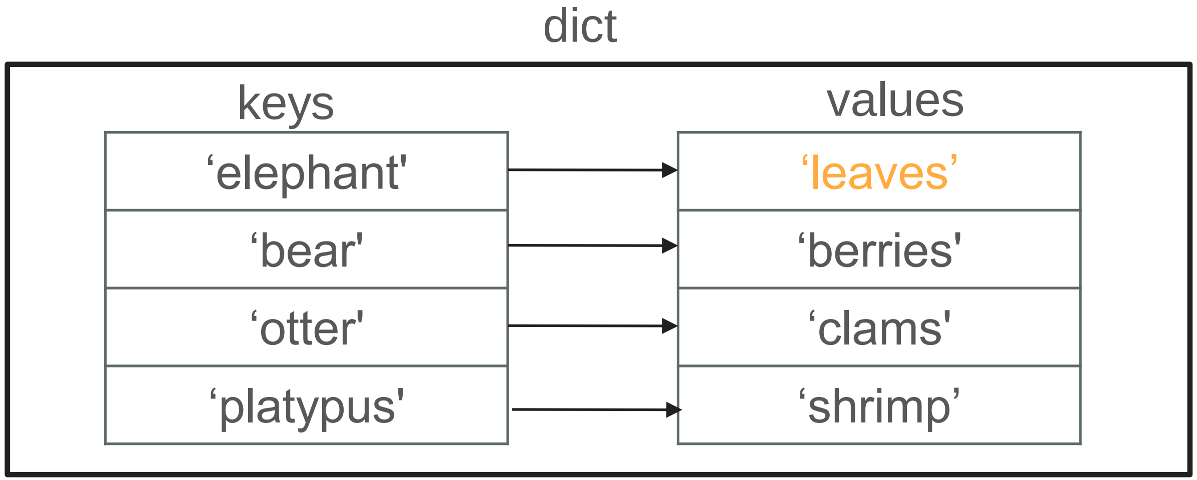
Dictionary - **in**

```
>>>'elephant' in d
```



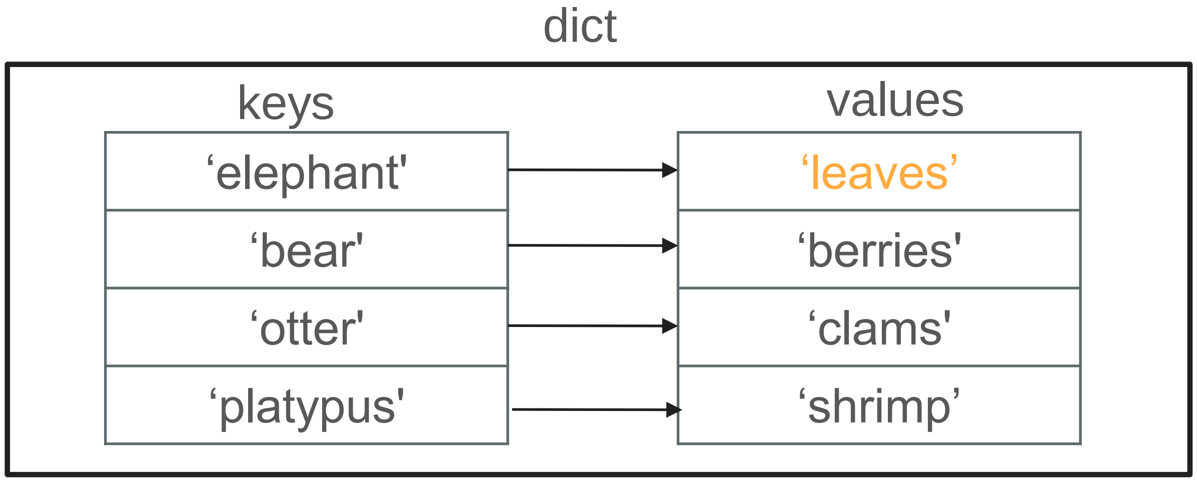
Dictionary - **in**

```
>>>'elephant' in d
True
```



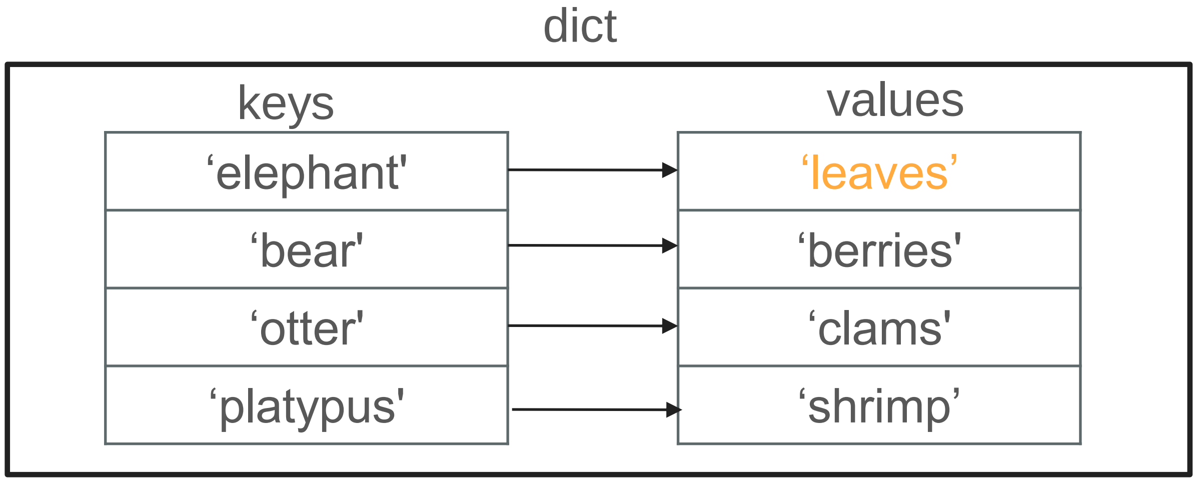
Dictionary - **in**

```
>>>'elephant' in d
True
>>>'cat' not in d
True
```



Dictionary - **in**

```
>>>'elephant' in d
True
>>>'cat' 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[‘elephant’] = ‘grass’
```

Building a dictionary

```
>>> d = {}

>>> d['elephant'] = 'grass'
```

We can add keys using "set"

Building a dictionary

```
>>> d = {}

>>> d['elephant'] = 'grass'

>>> d
```

We can add keys using "set"

Building a dictionary

```
>>> d = {}

>>> d['elephant'] = 'grass'

>>> d
{'elephant': 'grass'}
```

We can add keys using "set"

Building a dictionary

```
>>> d = {‘elephant’: ‘grass’}
```

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...

Think/Share:

Store names of CS lecturers and their ages

Accessing a Dictionary’s Keys

```
>>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
```

Building a dictionary

```
>>> d = {‘Buket’: 31}

>>> d[‘Buket’] += 2
```

Building a dictionary

```
>>> d = {‘Buket’: 31}

>>> d[‘Buket’] += 2
```

we can get/set on the same line!
(same as d[‘Buket’] = d[‘Buket’] + 2)

Building a dictionary

```
>>> d = {‘Buket’: 31}

>>> d[‘Buket’] += 2

>>> d[‘Buket’]
{‘Buket’: 33}
```

we can get/set on the same line!

(same as d[‘Buket’] = d[‘Buket’] + 2)

Accessing a Dictionary’s Keys

```
>>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}

>>> d.keys()
```

Accessing a Dictionary’s Keys

```
>>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}

>>> d.keys()
dict_keys(['Buket', ‘Nick’, 'Baris'])
```

Iterable collection of all the keys.

Iterable means it can be used in foreach

Accessing a Dictionary’s Keys

```
>> d = {‘Buket’: 31, 'Nick': 28, 'Baris':35}

>>> list(d.keys())
[‘Buket’, ‘Nick’, 'Baris']
```

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

Accessing a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
```

Accessing a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}

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

Accessing a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> list(d.values())
```

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

Accessing a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> list(d.values())
[31,28,35]
```

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

Looping over a Dictionary's Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
```

Looping over a Dictionary’s Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name in d.keys():
```

Looping over a Dictionary’s Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name in d.keys():
...   print(name)
```

Looping over a Dictionary’s Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name in d.keys():
...   print(name)

Buket
Nick
Baris
```

Looping over a Dictionary’s Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name in d.keys():
...     print(name)

Buket
Nick
Baris
```

we can use foreach on the dictionary’s keys!

Looping over a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
```

Looping over a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for age in d.values():
```

Looping over a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for age in d.values():
...   print(age)
```

Looping over a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}

>>> for age in d.values():
...    print(age)
31
28
35
```

Looping over a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for age in d.values():
...     print(age)
31
28
35
```

we can use foreach on the dictionary’s values!

Looping over a Dictionary’s Keys and Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
```

Looping over a Dictionary’s Keys and Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name, age in d.items():
```

Looping over a Dictionary’s Keys and Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name, age in d.items():
```

items() gives us key, value pairs

Looping over a Dictionary’s Keys and Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name, age in d.items():
...   print(name, ‘is’, age, ‘years old.’)
```

items() gives us key, value pairs

Looping over a Dictionary's Keys and Values

```
>> d = {'Buket': 31, 'Nick': 28, 'Baris': 35} >>>
 for name, age in d.items():
...    print(name, 'is', age, 'years old.')
Buket is 31 years old.
Nick is 28 years old.
Baris is 35 years old.
```

items() gives us key, value pairs

Looping over a Dictionary’s Keys and Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}

>>> for name, age in d.items():
...     print(name, ‘is’, age, ‘years old.’)

Buket is 31 years old.
Nick is 28 years old.
Baris is 35 years old.
.
```

print() will automatically concatenate args separated by commas!

Printing with sep=

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name, age in d.items():
...   print(name, age, sep=‘: ’)
```

Printing with sep=

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name, age in d.items():
...   print(name, age, sep=‘: ’)
```

sep is an optional argument like end!

Printing with sep=

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name, age in d.items():
...   print(name, age, sep=‘: ’)

Buket: 34
Nick: 28
Baris: 35
```

sep is an optional argument like end!

Printing with sep=

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> for name, age in d.items():
...   print(name, age, sep=‘: ’)
Buket: 34
Nick: 28
Baris: 35
```

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

Printing with sep=

```
>> d = {'Buket': 31, 'Nick': 28, 'Baris': 35}
>>> for name, age in d.items():
...    print(name, age, sep=': ')
Buket: 34
Nick: 28
Baris: 35
```

the default is sep=' ' (insert space)

Getting a Sorted List of Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
```

Getting a Sorted List of Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> sorted(d.keys())
```

Getting a Sorted List of Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> sorted(d.keys())
[‘Baris’,‘Buket’, ‘Nick’]
```

Getting a Sorted List of Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> sorted(d.keys())
[‘Buket’, ‘Nick’, ‘Baris’]
```

sorted() returns a list in alphabetical order!

Getting a Sorted List of Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> sorted(d.keys())
[‘Baris’, ‘Buket’, ‘Nick’]
>>> d
{'Buket': 31, 'Nick': 28, 'Baris': 35}
```

Getting a Sorted List of Keys

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> sorted(d.keys())
[‘Baris’, ‘Buket’, ‘Nick’]
```

Sorting a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
```

Sorting a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> sorted(d.values())
```

Sorting a Dictionary’s Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> sorted(d.values())
[28, 31, 35]
```

Sorting a Dictionary's Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> sorted(d.values())
[28, 31, 35]
```

sorted() returns a list in numerical order!

Retrieving Min/Max Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
```

Retrieving Min/Max Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> min(d.values())
```

Retrieving Min/Max Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> min(d.values())
```

returns the smallest element!

Retrieving Min/Max Values

```
>> d = {‘Buket’: 31, ‘Nick’: 28, ‘Baris’: 35}
>>> min(d.values())
28
```

returns the smallest element!

Retrieving Min/Max Values

```
>> d = {'Buket': 31, 'Nick': 28, 'Baris': 35}
>>> min(d.values())
28
>>> max(d.values())
```

returns the smallest element!

Retrieving Min/Max Values

```
>> d = {'Buket': 31, 'Nick': 28, 'Baris': 35}
>>> min(d.values())
28
>>> max(d.values())
```

returns the smallest element!

returns the biggest element!

Retrieving Min/Max Values

```
>> d = {'Buket': 31, 'Nick': 28, 'Baris': 35}
>>> min(d.values())
28
>>> max(d.values())
35
```

returns the smallest element!

returns the biggest element!

List & dictionary operations

	Lists	Dictionaries
A new empty variable	my_list = [ ]	my_dict = { }
A new variable with values	my_list = [1, 2, 3]	my_dict = {k1:v1, k2:v2, k3:v3}
Accessing an entry	my_list[indx]	my_dict[key]
Adding an item	my_list.append(item)	my_dict[key] = value
Changing the value of an item	my_list[indx] = new_value	my_dict[key] = new_value
Removing an item	my_list.pop(indx)	my_dict.pop(key)

What’s next?

Nested Data Structures

- We can nest data structures!
  - Lists in lists
    - *grid/game board*
  - Lists in dicts
    - *animals to feeding times*
  - Dicts in dicts
    - *your phone's contact book*
  - ... and so on!

How to organize data

Ice cream sales

	June	**July**	**August**
2018	500	700	600
2019	550	750	700
2020	250	500	400

How to organize data

Ice cream sales

	June	**July**	**August**
2018	500 ice[0][0]	700 ice[0][1]	600 ice[0][2]
2019	550 ice[1][0]	750 ice[1][1]	700 ice[1][2]
2020	250 ice[2][0]	500 ice[2][1]	400 ice[2][2]

```
ice = [[500,700,600], [550,750,700], [250,500,400]]
```

Example: June 2020 ice cream sales is accessed as `ice[2][0]`

How to organize data

Ice cream sales

	June	July	August
2018	500	700	600
2019	550	750	700
2020	250	500	400

```
ice = {2018: {'june':500, 'july':700, 'august':600},
       2019: {'june':500, 'july':700, 'august':600},
       2020: {'june':500, 'july':700, 'august':600}}
```

Example: June 2020 ice cream sales is accessed as `ice[2020]['june']`

Think/Share:

Implement a phone book using dictionaries