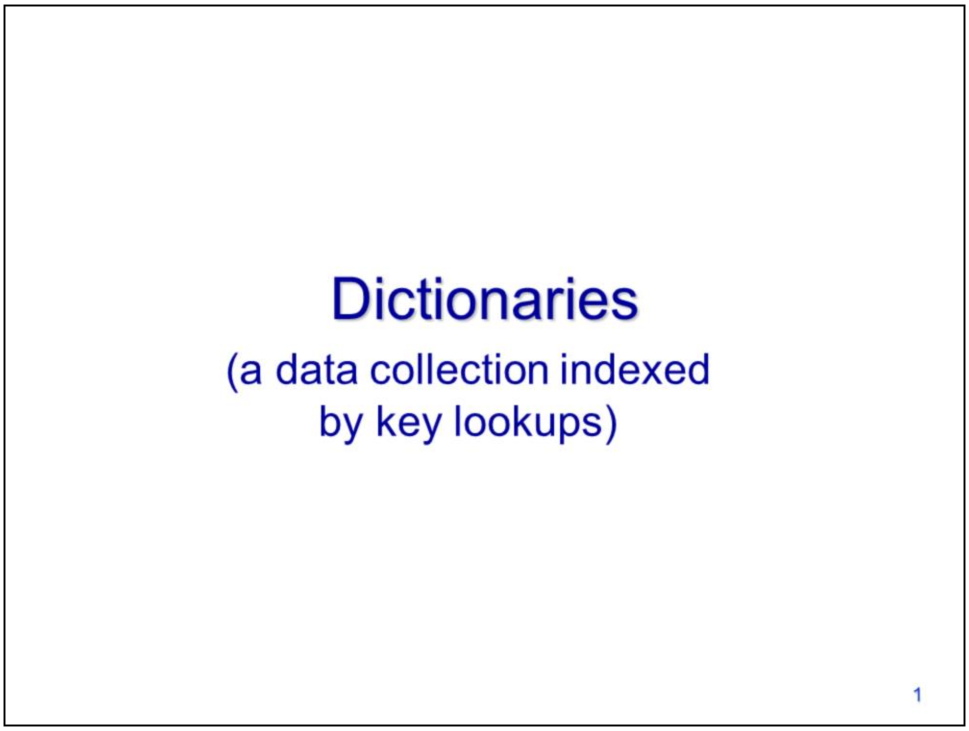
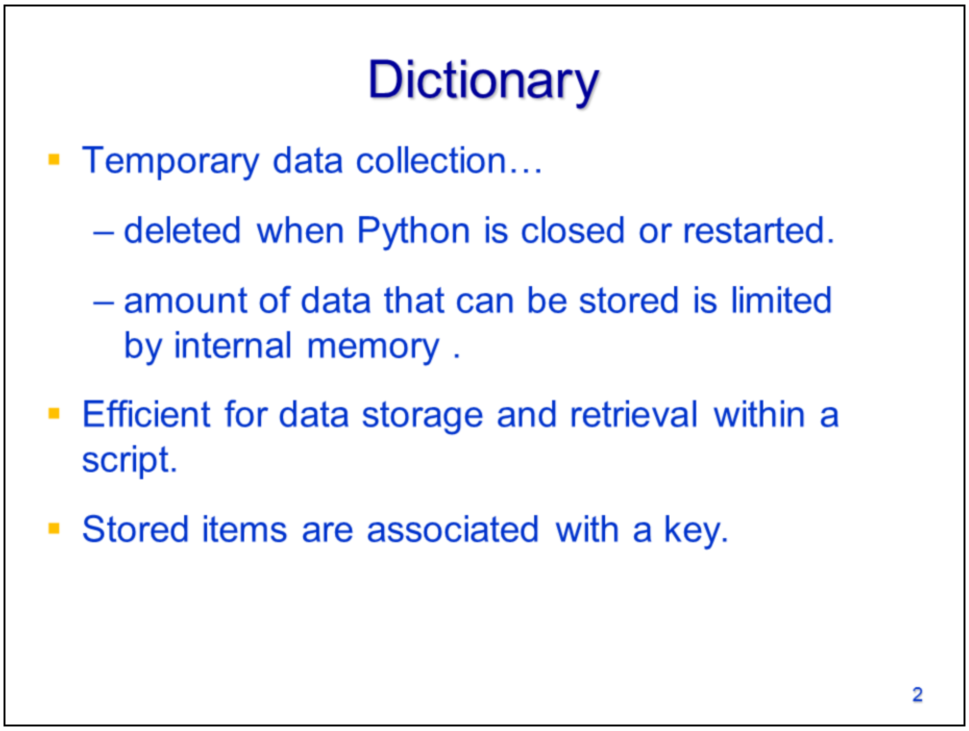


Dictionaries

(a data collection indexed by key lookups)

1

This video will discuss Python dictionaries.

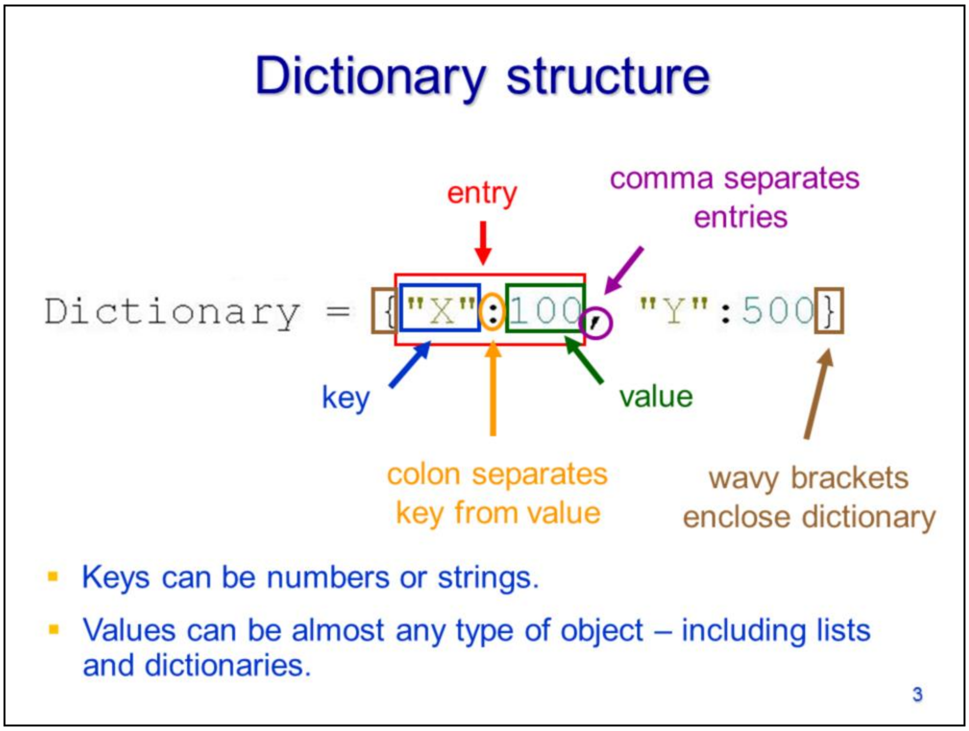


Dictionary

- Temporary data collection…
  - – deleted when Python is closed or restarted.
  - – amount of data that can be stored is limited by internal memory .
- Efficient for data storage and retrieval within a script.
- Stored items are associated with a key.

2

Dictionaries are similar to lists in that they are used for temporary storage and retrieval of data within a script. Items in a dictionary are associated with keys – an item is retrieved by specifying the key that it is associated with. The decision of whether to use a list or dictionary is based primarily on convenience.



Dictionary structure

```
Dictionary = {"X":100, "Y":500}
```

entry
comma separates entries
key
value
colon separates key from value
wavy brackets enclose dictionary

- Keys can be numbers or strings.
- Values can be almost any type of object – including lists and dictionaries.

3

Here we see an example of a Python dictionary.

It is enclosed by wavy brackets…

and contains **entries**.

Each entry contains a **key…**

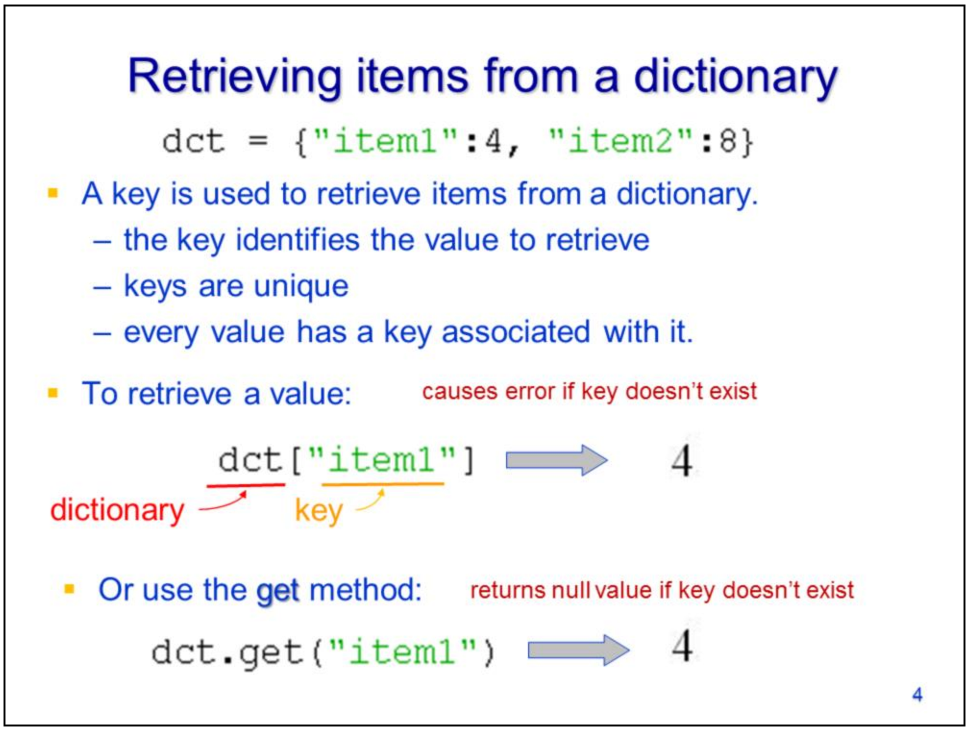
And its associated **value**.

The key is separated from the value by a colon.

Entries are separated by commas.

Keys can only be numbers or strings.

Values can be any type of object including lists and dictionaries.



Retrieving items from a dictionary

```
dct = {"item1":4, "item2":8}
```

- A key is used to retrieve items from a dictionary.
  - the key identifies the value to retrieve
  - keys are unique
  - every value has a key associated with it.
- To retrieve a value: causes error if key doesn't exist

$\underbrace{\texttt{dct}}_{\text{dictionary}}\texttt{[}\underbrace{\texttt{"item1"}}_{\text{key}}\texttt{]} \Longrightarrow 4$

- Or use the **get** method: returns null value if key doesn't exist

```
dct.get("item1")
```
$\Longrightarrow 4$

4

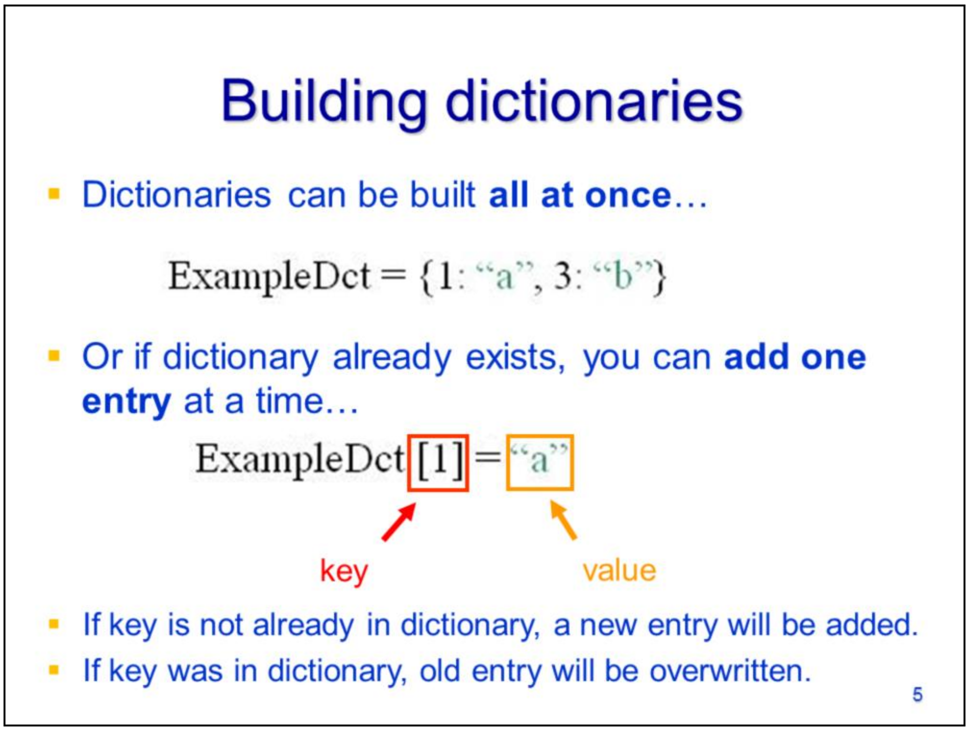
Items are retrieved from a dictionary by specifying the key that is associated with the item. Each key in a given dictionary must be unique.

To retrieve an item from a dictionary, specify the dictionary name followed by the appropriate key enclosed in square brackets.

In this case, the “item1” key is associated with the integer 4.

An alternative method of retrieving items is to use the dictionary’s **get** method.

Both methods are equivalent except that the get method will not crash the script if you specify a key that does not exist.



Building dictionaries

- Dictionaries can be built **all at once**…

ExampleDct = {1: “a”, 3: “b”}

- Or if dictionary already exists, you can **add one entry** at a time…

ExampleDct $\boxed{[1]}$ = $\boxed{\text{“a”}}$

- If key is not already in dictionary, a new entry will be added.
- If key was in dictionary, old entry will be overwritten.

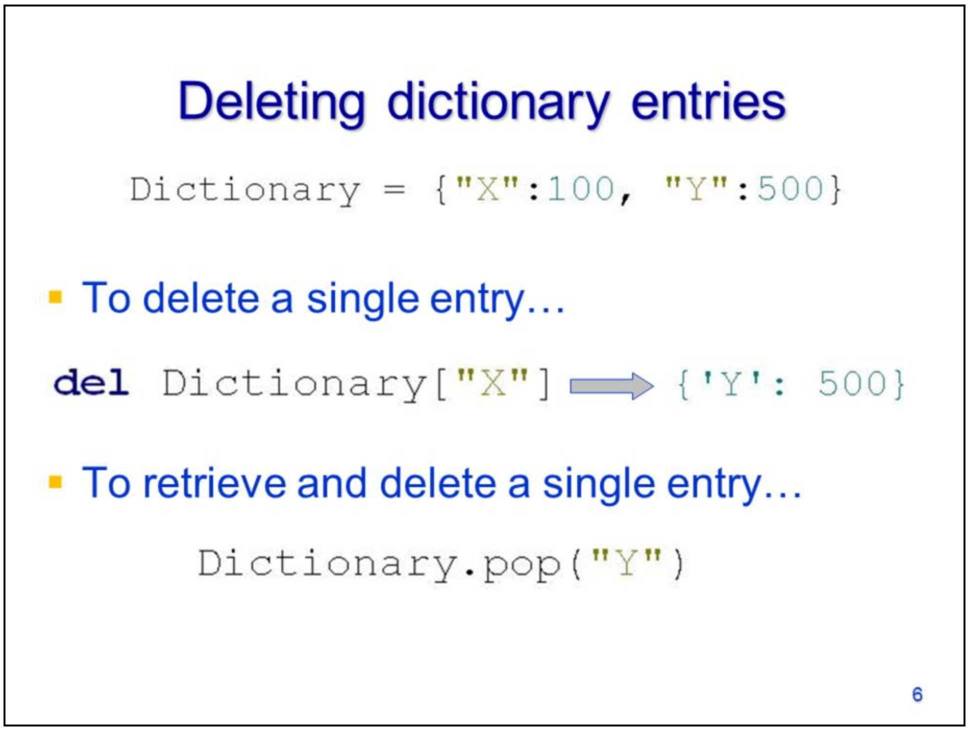
5

Entries can be added to a dictionary when the are first created.

Entries can also be added to existing dictionaries.

If the key does not already exist in the dictionary, then a new entry will be created.

If the key does exist in the dictionary, then the new value will overwrite the previous value associated with the key.



Deleting dictionary entries

```
Dictionary = {"X":100, "Y":500}
```

- To delete a single entry…

```
del Dictionary["X"]
```
$\Longrightarrow$
```
{'Y': 500}
```

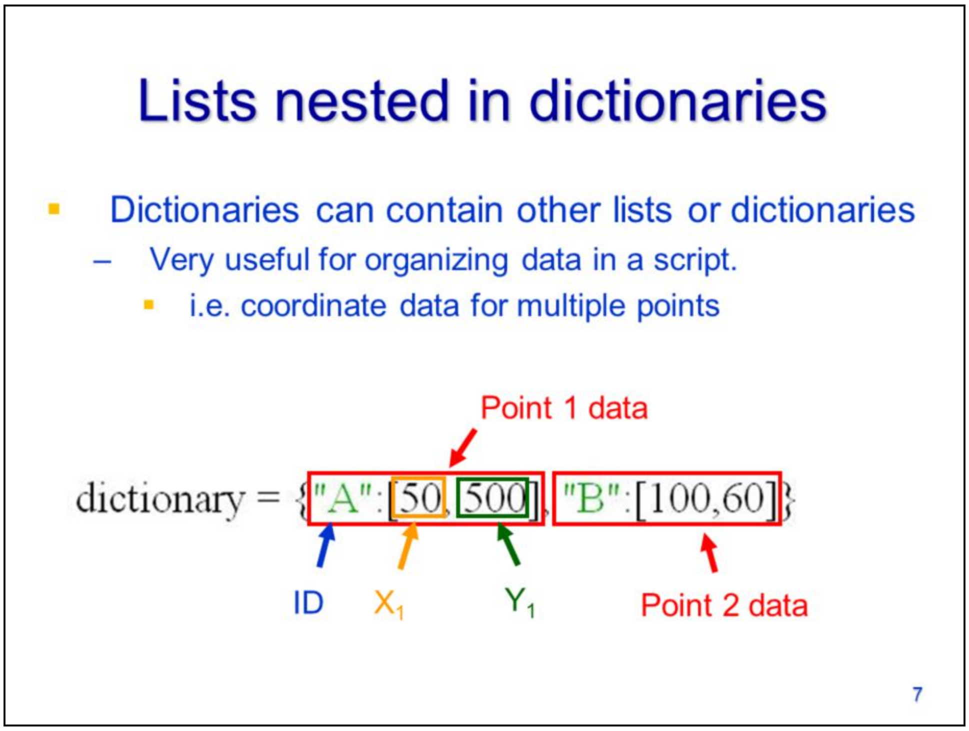
- To retrieve and delete a single entry…

```
Dictionary.pop("Y")
```

6

Entries can be removed from a dictionary by using the **del** command and specifying the key associated with the entry.

The dictionary’s **pop** method will also delete items from a dictionary as well as return the value associated with the key.



Lists nested in dictionaries

- Dictionaries can contain other lists or dictionaries
  - Very useful for organizing data in a script.
    - i.e. coordinate data for multiple points

```
dictionary = {"A":[50,500],"B":[100,60]}
```

Point 1 data
ID
X_1
Y_1
Point 2 data

7

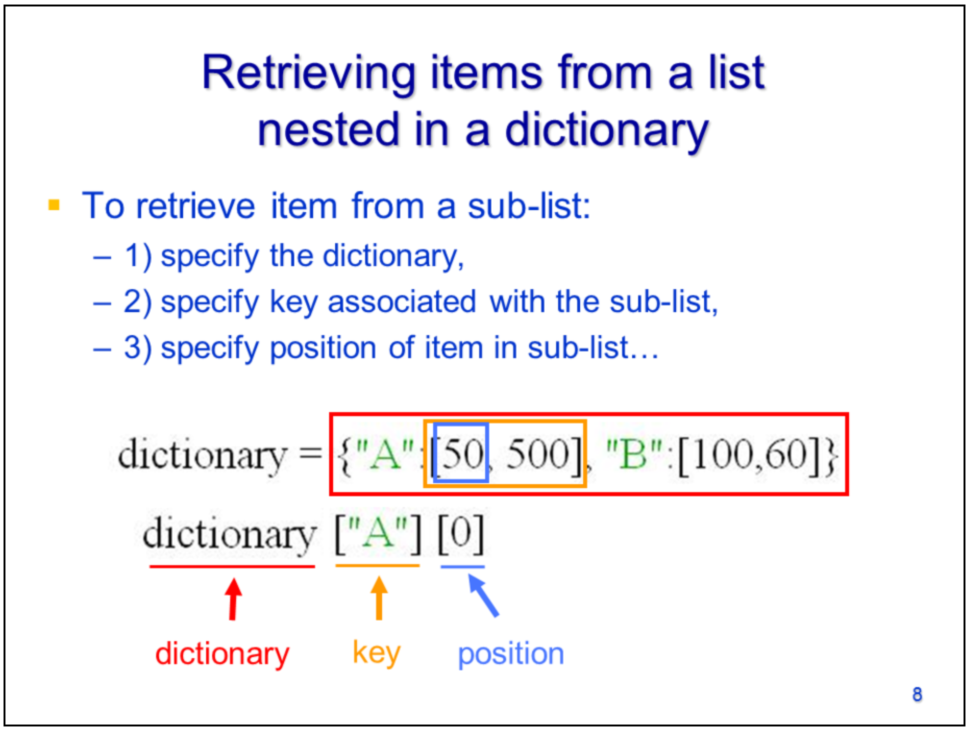
Dictionaries can contain lists or other dictionaries. “Nesting” lists or dictionaries can be useful for organizing data. In this example, we’ll see how a dictionary can be used to store point coordinates.

The dictionary contains the full set of points.

Each entry in the dictionary contains the data for a single point.

The key for each entry is the ID for the point

The associated value is a list containing the X and Y coordinates for the point.



Retrieving items from a list nested in a dictionary

- To retrieve item from a sub-list:
  - 1) specify the dictionary,
  - 2) specify key associated with the sub-list,
  - 3) specify position of item in sub-list…

```
dictionary = {"A":[50, 500], "B":[100,60]}
dictionary ["A"] [0]
```

dictionary
key
position

8

This example will show how to retrieve an item from a list that is nested in a dictionary.

Specify the name of the dictionary…

Followed by the key associated with the list containing the item…

Followed by the position of the item in the list.