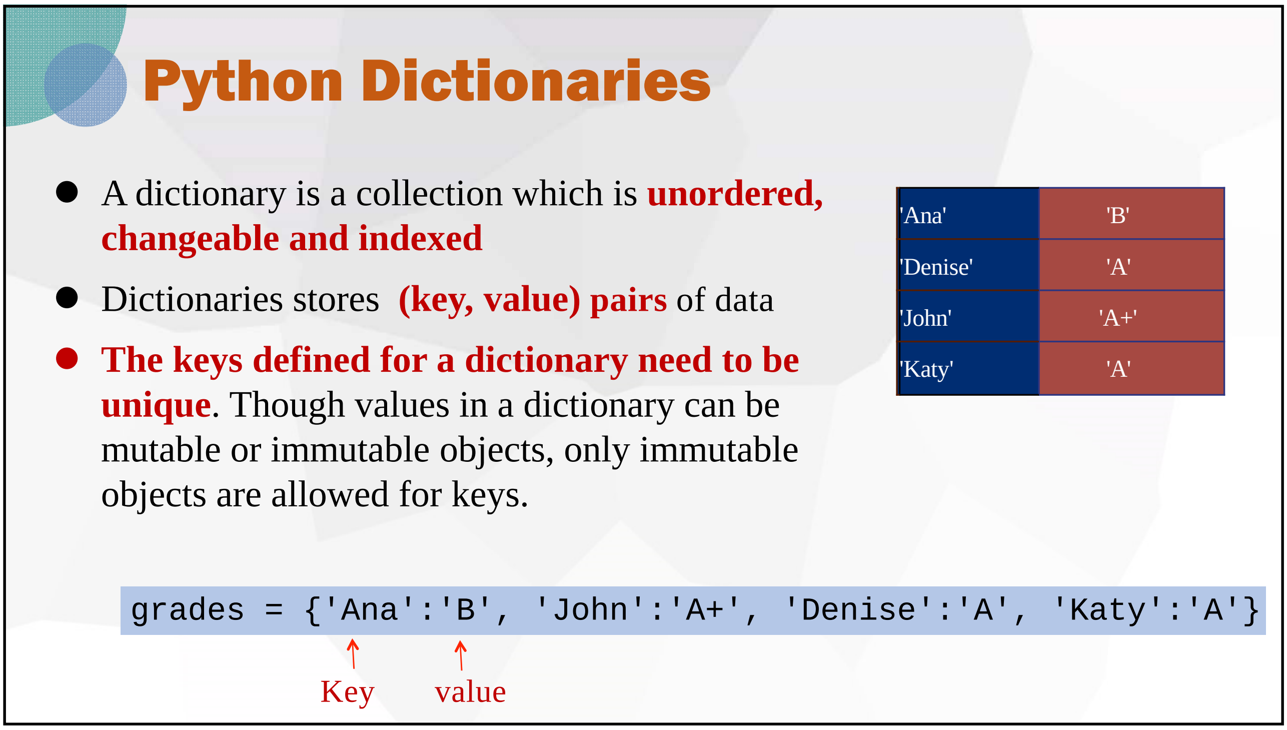


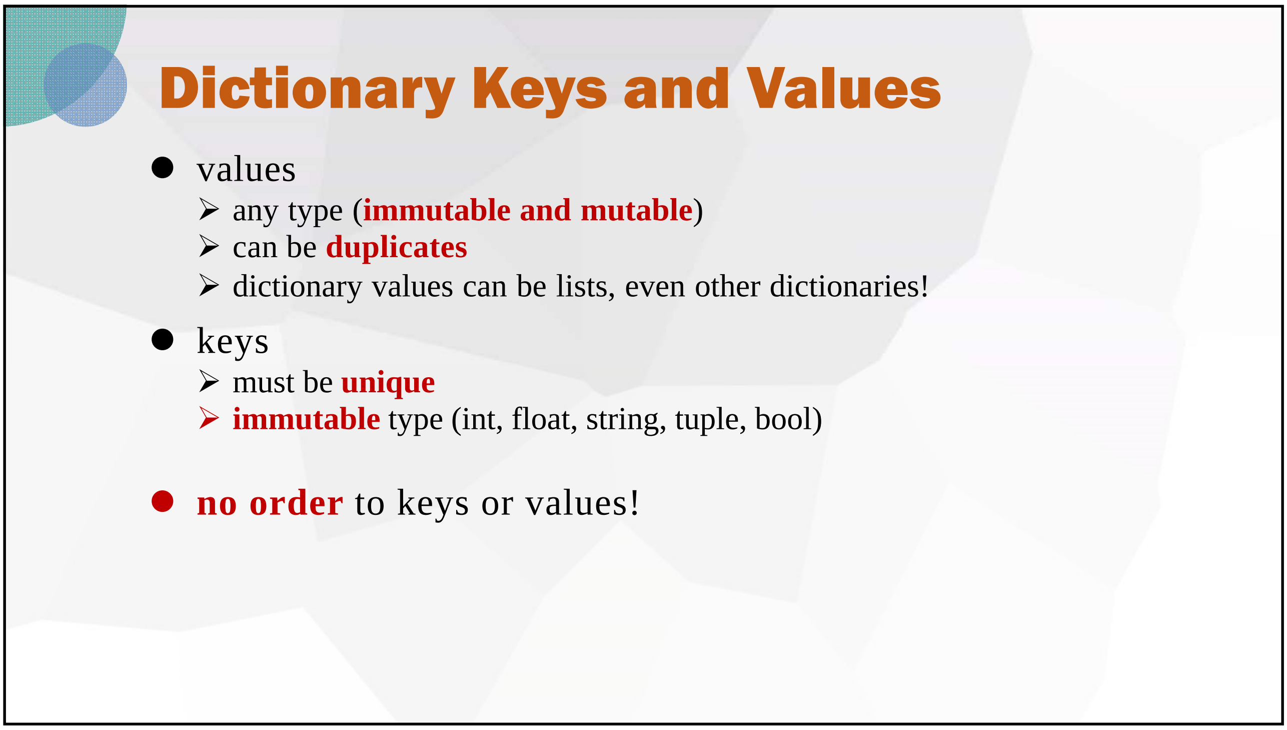
Python Dictionaries

- A dictionary is a collection which is **unordered, changeable and indexed**
- Dictionaries stores **(key, value) pairs** of data
- **The keys defined for a dictionary need to be unique**. Though values in a dictionary can be mutable or immutable objects, only immutable objects are allowed for keys.

'Ana'	'B'
'Denise'	'A'
'John'	'A+'
'Katy'	'A'

```
grades = {'Ana':'B', 'John':'A+', 'Denise':'A', 'Katy':'A'}
```

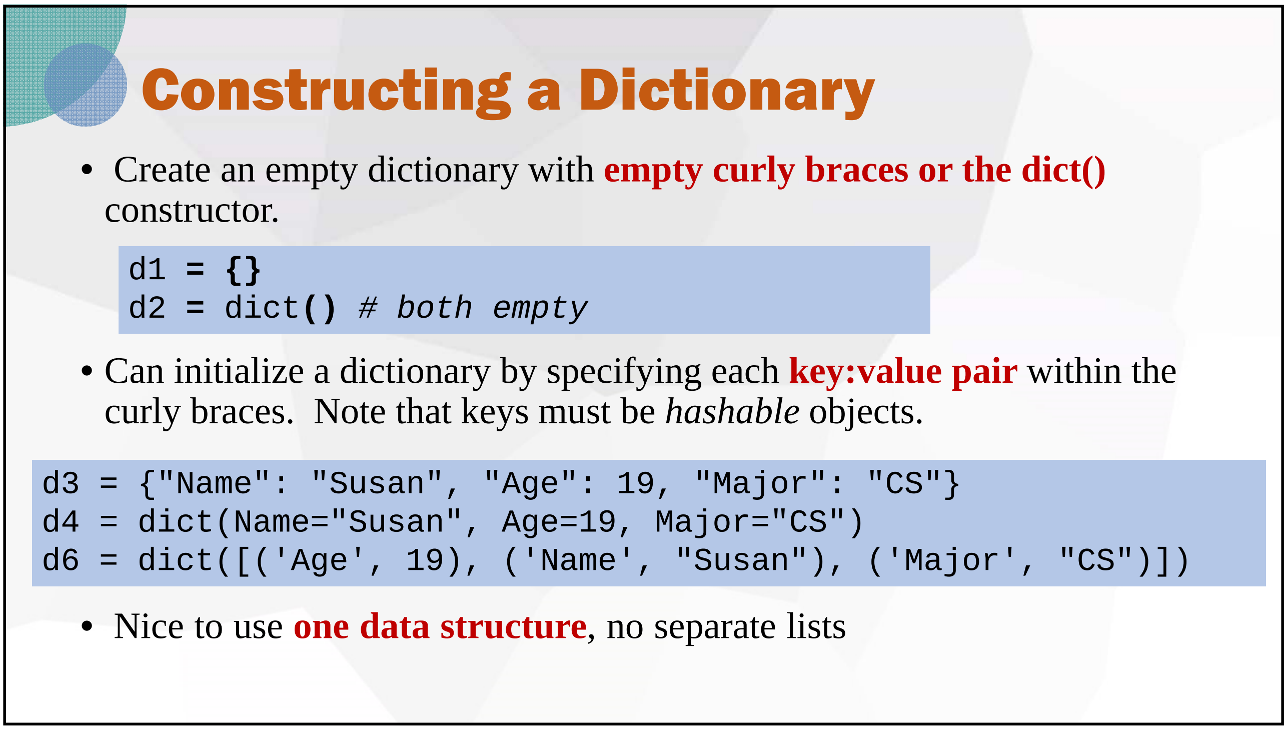
↑ Key ↑ value



Dictionary Keys and Values

- values
  - any type (**immutable and mutable**)
  - can be **duplicates**
  - dictionary values can be lists, even other dictionaries!
- keys
  - must be **unique**
  - **immutable** type (int, float, string, tuple, bool)

- **no order** to keys or values!



Constructing a Dictionary

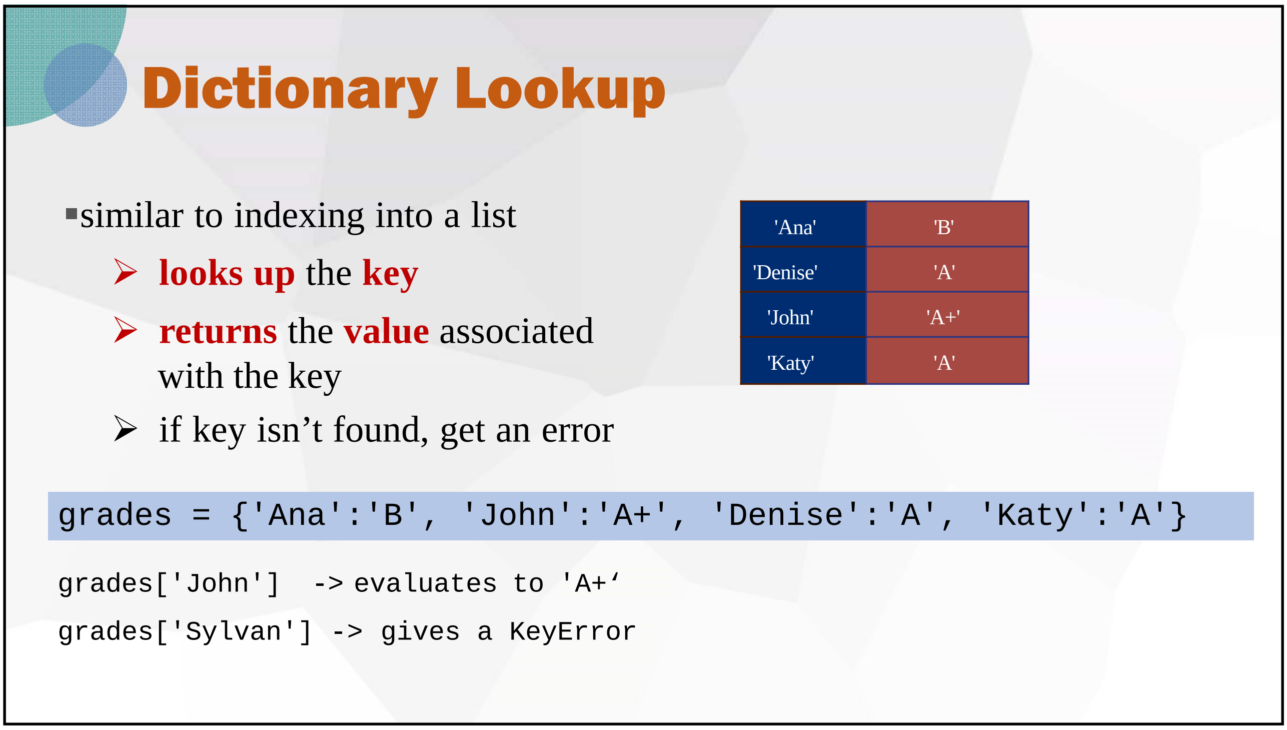
- Create an empty dictionary with **empty curly braces or the dict()** constructor.

```
d1 = {}
d2 = dict() # both empty
```

- Can initialize a dictionary by specifying each **key:value pair** within the curly braces. Note that keys must be *hashable* objects.

```
d3 = {"Name": "Susan", "Age": 19, "Major": "CS"}
d4 = dict(Name="Susan", Age=19, Major="CS")
d6 = dict([('Age', 19), ('Name', "Susan"), ('Major', "CS")])
```

- Nice to use **one data structure**, no separate lists



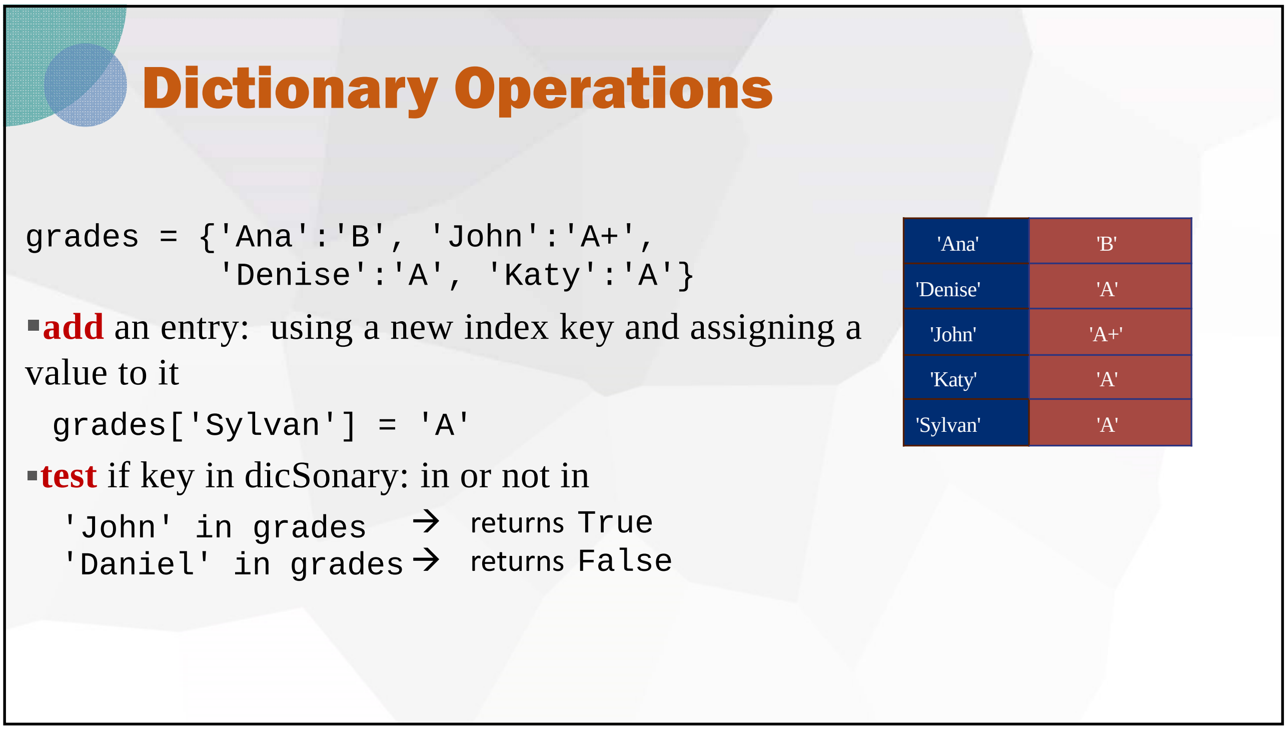
Dictionary Lookup

- similar to indexing into a list
  - **looks up** the **key**
  - **returns** the **value** associated with the key
  - if key isn’t found, get an error

'Ana'	'B'
'Denise'	'A'
'John'	'A+'
'Katy'	'A'

```
grades = {'Ana':'B', 'John':'A+', 'Denise':'A', 'Katy':'A'}
```

```
grades['John']  -> evaluates to 'A+‘
grades['Sylvan'] -> gives a KeyError
```



Dictionary Operations

```
grades = {'Ana':'B', 'John':'A+',
          'Denise':'A', 'Katy':'A'}
```

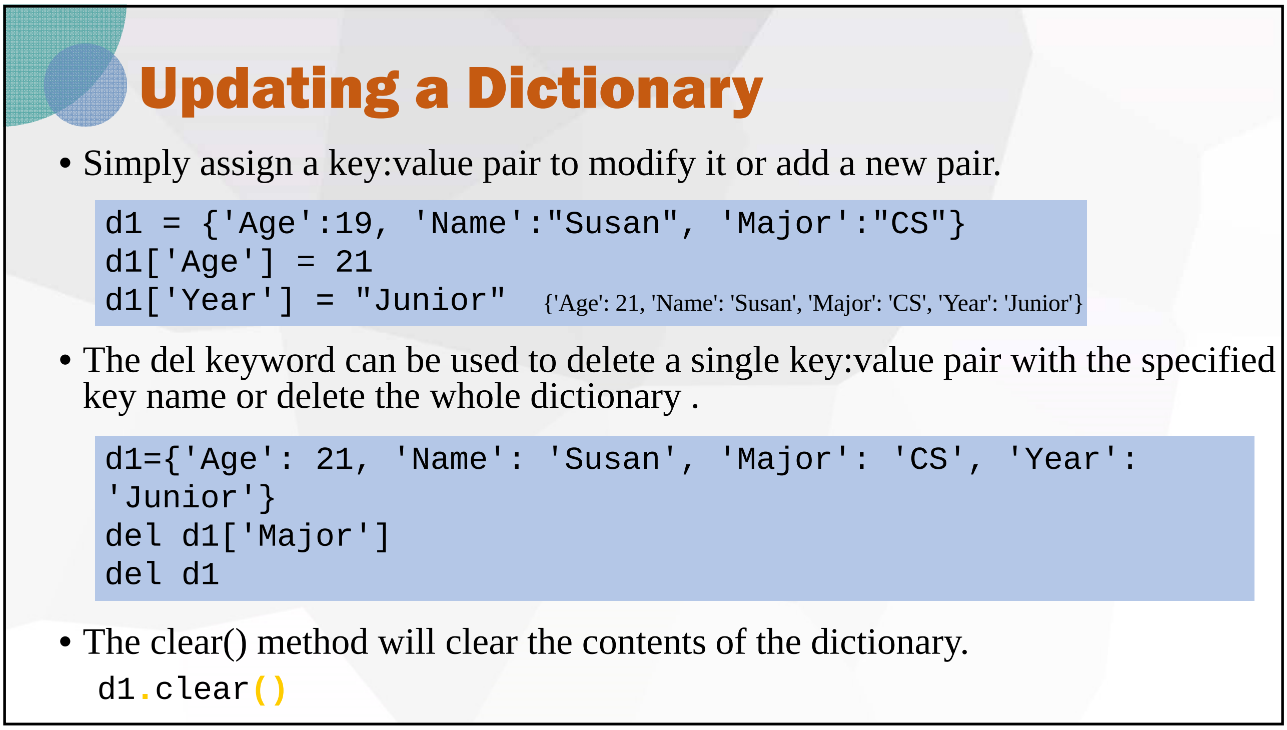
- **add** an entry: using a new index key and assigning a value to it

```
 grades['Sylvan'] = 'A'
```

- **test** if key in dicSonary: in or not in

```
  'John' in grades    →  returns True
  'Daniel' in grades →  returns False
```

'Ana'	'B'
'Denise'	'A'
'John'	'A+'
'Katy'	'A'
'Sylvan'	'A'



Updating a Dictionary

- Simply assign a key:value pair to modify it or add a new pair.

```
d1 = {'Age':19, 'Name':"Susan", 'Major':"CS"}
d1['Age'] = 21
d1['Year'] = "Junior"
```

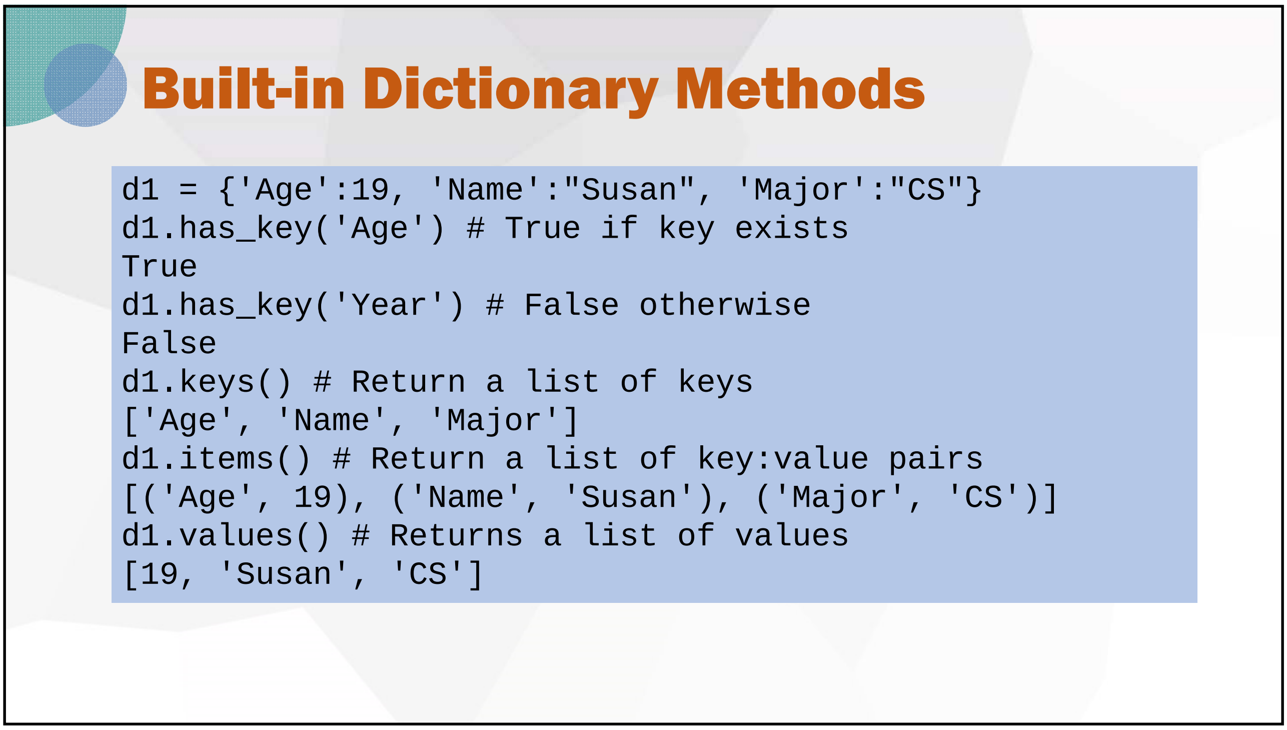
```
{'Age': 21, 'Name': 'Susan', 'Major': 'CS', 'Year': 'Junior'}
```

- The del keyword can be used to delete a single key:value pair with the specified key name or delete the whole dictionary .

```
d1={'Age': 21, 'Name': 'Susan', 'Major': 'CS', 'Year':
'Junior'}
del d1['Major']
del d1
```

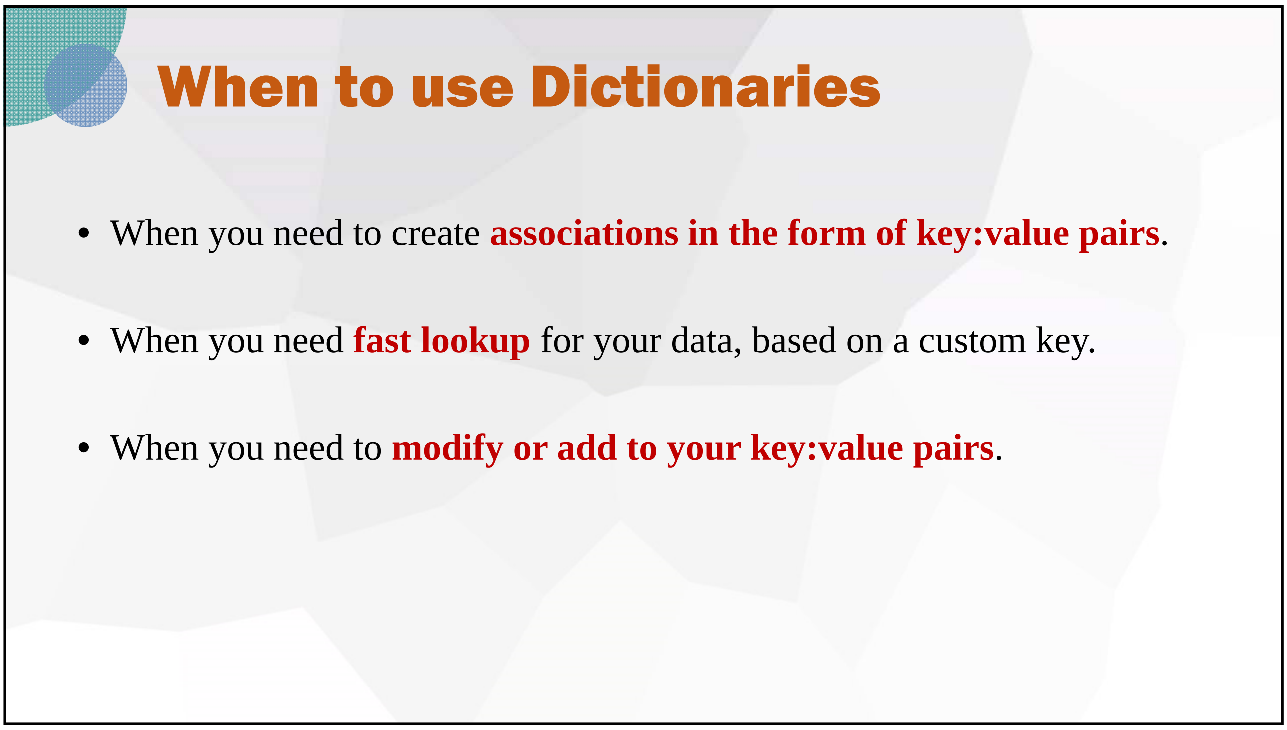
- The clear() method will clear the contents of the dictionary.

```
d1.clear()
```



Built-in Dictionary Methods

```
d1 = {'Age':19, 'Name':"Susan", 'Major':"CS"}
d1.has_key('Age') # True if key exists
True
d1.has_key('Year') # False otherwise
False
d1.keys() # Return a list of keys
['Age', 'Name', 'Major']
d1.items() # Return a list of key:value pairs
[('Age', 19), ('Name', 'Susan'), ('Major', 'CS')]
d1.values() # Returns a list of values
[19, 'Susan', 'CS']
```



When to use Dictionaries

- When you need to create **associations in the form of key:value pairs**.
- When you need **fast lookup** for your data, based on a custom key.
- When you need to **modify or add to your key:value pairs**.