Mini-Lecture 15

Dictionaries

Dictionaries (Type dict)

Description

- List of key-value pairs
  - Keys are unique
  - Values need not be
- Example: net-ids
  - net-ids are **unique** (a key)
  - names need not be (values)
  - js1 is John Smith (class '13)
  - js2 is John Smith (class '16)
- Many other applications

Python Syntax

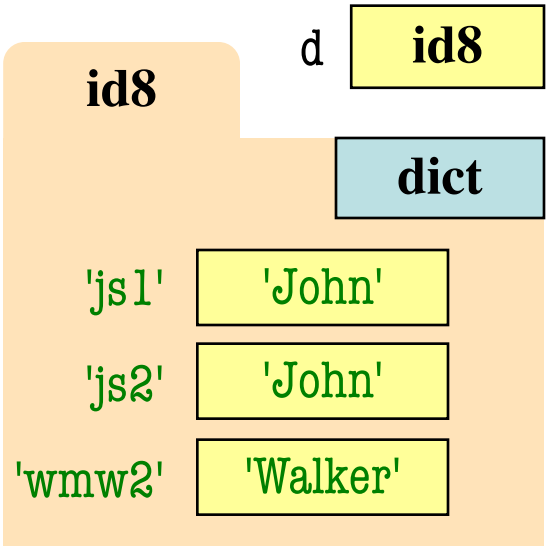
- Create with format:
  `{k1:v1, k2:v2, ...}`
- Keys must be non-mutable
  - ints, floats, bools, strings
  - **Not** lists or custom objects
- Values can be anything
- Example:

```
d = {'js1':'John Smith',
     'js2':'John Smith',
     'wmw2':'Walker White'}
```

Using Dictionaries (Type **dict**)

- Access elts. like a list
  - d['js1'] evaluates to 'John'
  - But cannot slice ranges!
- Dictionaries are **mutable**
  - Can reassign values
  - d['js1'] = 'Jane'
  - Can add new keys
  - d['aa1'] = 'Allen'
  - Can delete keys
  - del d['wmw2']

```
d = {'js1':'John','js2':'John',
     'wmw2':'Walker'}
```

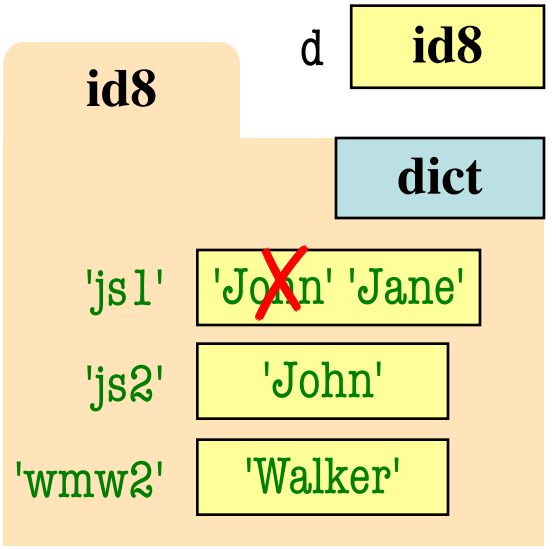


Key-Value order in folder is not important

Using Dictionaries (Type **dict**)

- Access elts. like a list
  - d['js1'] evaluates to 'John'
  - But cannot slice ranges!
- Dictionaries are **mutable**
  - Can reassign values
  - d['js1'] = 'Jane'
  - Can add new keys
  - d['aa1'] = 'Allen'
  - Can delete keys
  - del d['wmw2']

```
d = {'js1':'John','js2':'John',
     'wmw2':'Walker'}
```

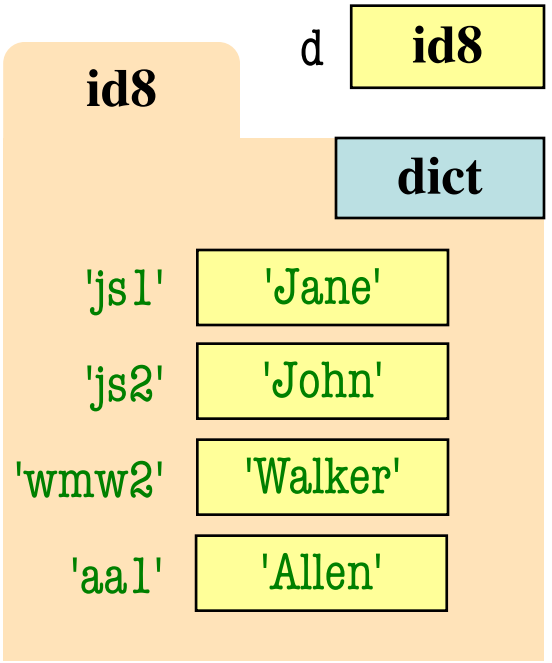


Key-Value order in folder is not important

Using Dictionaries (Type **dict**)

- Access elts. like a list
  - d['js1'] evaluates to 'John'
  - But cannot slice ranges!
- Dictionaries are **mutable**
  - Can reassign values
  - d['js1'] = 'Jane'
  - Can add new keys
  - d['aa1'] = 'Allen'
  - Can delete keys
  - del d['wmw2']

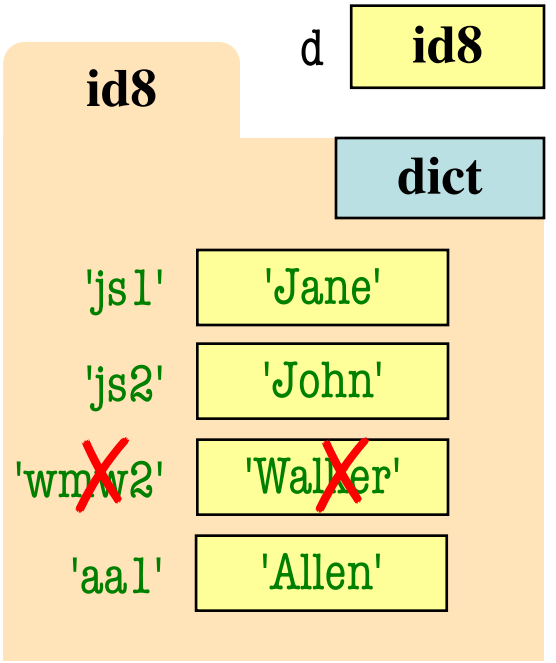
```
d = {'js1':'John','js2':'John',
 'wmw2':'Walker'}
```



Using Dictionaries (Type **dict**)

- Access elts. like a list
  - d['js1'] evaluates to 'John'
  - But cannot slice ranges!
- Dictionaries are **mutable**
  - Can reassign values
  - d['js1'] = 'Jane'
  - Can add new keys
  - d['aa1'] = 'Allen'
  - Can delete keys
  - del d['wmw2']

```
d = {'js1':'John','js2':'John',
     'wmw2':'Walker'}
```



Deleting key deletes both

Dictionaries and For-Loops

- Dictionaries != sequences
  - Cannot slice them
- *Different* inside for loop
  - Loop variable gets the key
  - Then use key to get value
- Can **extract iterators** with dictionary *methods*
  - Key iterator: d.keys()
  - Value iterator: d.values()
  - key-value pairs: d.items()

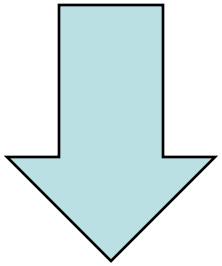
```
for k in d:
    # Loops over keys
    print(k)      # key
    print(d[k])   # value


# To loop over values only
for v in d.values():
    print(v)      # value
```

See grades.py

Remembering Assignment 1

- JSONs are strings that look like dictionaries!

```
'{ "src" : "2.5 United States Dollars", '+
' "dst" : "2.15 Euros", "valid" : true, '+
' "error" : "" }'
```



```
{ "src" : "2.5 United States Dollars",
  "dst" : "2.15 Euros",
  "valid" : True,
  "error" : ""}
```

Remembering Assignment 1

- JSONs are strings that look like dictionaries!

```
'{ "src" : "2.5 United States Dollars", '+
' "dst" : "2.15 Euros", "valid" : true, '+
' "e
```

json module allows us to convert JSON strings to dictionaries (and vice versa)

```
{ "src" : "2.5 United States Dollars",
  "dst" : "2.15 Euros",
  "valid" : True,
  "error" : ""}
```

Remembering Assignment 1

```
>>> import json
>>> import a1
>>> result = a1.currency_response('USD','EUR',2.5)
>>> data = json.loads(result)
>>> data['src']
'2.5 United States Dollars'
>>> data['valid']
True
```