Dr. Gunjan Mansingh
Department of Computing
UWI

July 4- 29, 2022

Week 2 - Day 5, Session 1

More Data Structures

We have seen the list data structure and its uses.

We will now examine another data structure, the *dictionary.*

What is a Dictionary?

In data structure terms, a dictionary is better termed an associative list.

You can think if it as a list of pairs, where the first element of the pair, the key, is used to retrieve the second element, the value.

Thus we associate a key to a value.

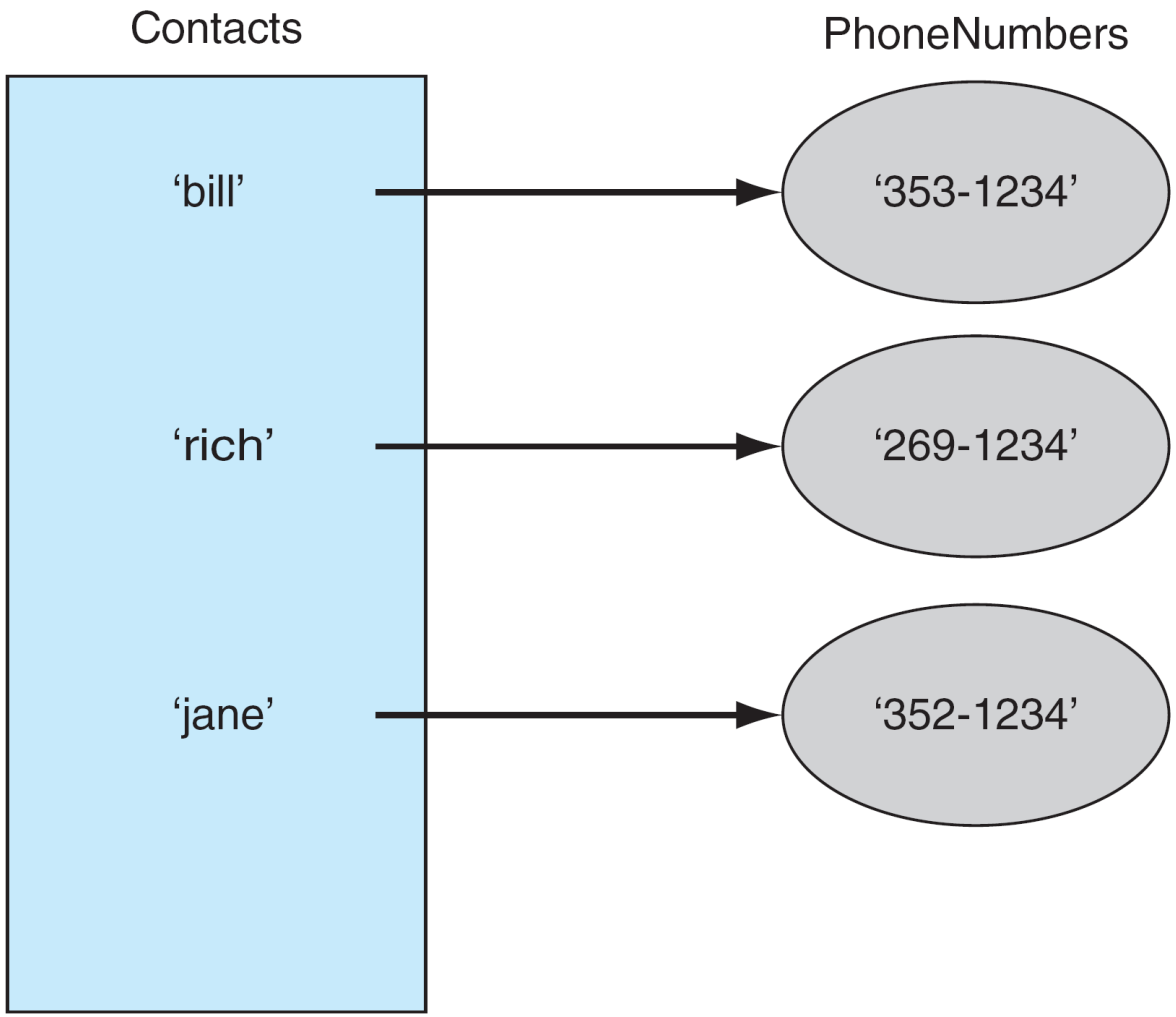
Python Dictionary

Use the { } marker to create a dictionary

Use the : marker to indicate key:value pairs:

```
contacts= {'bill': '353-1234',
'rich': '269-1234', 'jane': '352-1234'}
print contacts
{'jane': '352-1234',
'bill': '353-1234',
'rich': '369-1234'}
```

Example - Phone Contact List



Keys and Values

Key must be immutable:

- strings, integers, tuples are fine
- lists are NOT

Value can be anything.

Collections but not a Sequence

Dictionaries are collections, but they are not sequences like lists, strings or tuples:

- there is no order to the elements of a dictionary
- in fact, the order (for example, when printed) might change as elements are added or deleted.

So how to access dictionary elements?

Access Dictionary Elements

Access requires [ ], but the *key* is the index!

`myDict={}`

- an empty dictionary

`myDict['bill']=25`

- added the pair 'bill':25

`print(myDict['bill'])`

- prints `25`

Again, Common Operators

Like others, dictionaries respond to these:

`dict()`
- `Creates a dictionary`

`len(myDict)`
- number of key:value **pairs** in the dictionary

`<element> in myDict`
- boolean, is `element` a **key** in the dictionary

`for key in myDict:`
- iterates through the **keys** of a dictionary

Lots of Methods

`myDict.items()` – all the key/value pairs

`myDict.keys()` – all the keys

`myDict.values()` – all the values

`key in myDict` – does the key exist in the dictionary

`myDict.get(key,[default])` – takes a key and a value and returns the value if the key is found. If no value is supplied default is None.

`myDict.clear()` – empty the dictionary

`myDict.update(yourDict)` – for each key in `yourDict`, updates `myDict` with that key/value pair

Dictionaries are Iterable

```
for k in myDict:
    print(k)
```

- prints all the keys

```
for k,v in myDict.items():
    print (k,v)
```

- prints all the key/value pairs

```
for v in myDict.values():
    print (v)
```

- prints all the values

Building Dictionaries Faster

`zip` creates pairs from two parallel lists:

- `zip("abc",[1,2,3])` yields

`[('a',1),('b',2),('c',3)]`

That's good for building dictionaries. We call the `dict` function which takes a list of pairs to make a dictionary:

- `dict(zip("abc",[1,2,3]))` yields
- `{'a': 1, 'c': 3, 'b': 2}`

Examples

```
qp = {"A+” : 4.3, "A” : 4.0, "A-” : 3.7, "B+“ : 3.3, "B“ : 3.0, "B-“ : 2.7,
 "C+“ : 2.3, "C“ : 2.0, “F1“ : 1.7, “F2“ : 1.3, “F3“ : 0.0}

len(qp)

11

qp[“B+"]

3.3

"F1" in qp

True

"T" in qp

False
```

Examples

credit_list={'COMP1126':3,'COMP1127': 3, 'COMP1161':3, ‘COMP2101':3, 'COCR2003':1, 'COMP6101':6}

>>> for k in credit_list: print(k)

COCR2003

COMP1126

COMP6101

COMP1127

COMP1161

COMP2101

Examples

credit_list={'COMP1126':3,'COMP1127': 3, 'COMP1161':3, 'COMP2111':3, 'COMP2190':3, 'COMP2140':3,'COCR2003':1, 'COMP6101':6}

>>> for k,v in credit_list.items(): print(k,v)

COMP1126 3

COMP1161 3

COCR2003 1

COMP2111 3

COMP1127 3

COMP6101 6

COMP2190 3

COMP2140 3

Examples

```
credit_list={'COMP1126':3,'COMP1127': 3, 'COMP1161':3, 'COMP2111':3,  'COMP2190':3,
 'COMP2140':3,'COCR2003':1, 'COMP6101':6}

def find_credit(ccode):
    if ccode in credit_list:
        return credit_list[ccode]


find_credit('COMP1126')
3
find_credit('COMP6101')
6
```

Example
reverse_lookup(credit_list,3)

```
def reverse_lookup(d,v):
    for k in d:
        if d[k] == v:
            return k
```

Example
reverse_lookup_all(credit_list,3)

```
def reverse_lookup_all(d,v):
    val_list = []
    for k in d:
        if d[k] == v:
            val_list = val_list + [k]
    return val_list
```

Examples

credit_list={'COMP1126':3,'COMP1127': 3, 'COMP1161':3, 'COMP2111':3, 'COMP2190':3,‘ COMP2140':3,'COCR2003':1, 'COMP6101':6}

```
>>> reverse_lookup(credit_list,1)
'COCR2003‘
>>> reverse_lookup(credit_list,3)
'COMP2111‘
>>> reverse_lookup_all(credit_list,3)
['COMP1126', 'COMP1161', 'COMP2111', 'COMP1127', 'COMP2190',
'COMP2140']
```