Accelerating Information Technology Innovation

http://aiti.mit.edu

Nigeria Summer 2012
Lecture 4 – Data Structures

MISTI

Organizing the Football Universe

- Leagues -> Teams -> Many Players
- Teams and players can be represented by strings
- Build a data structure so that users can:
  - Check whether a team belongs to a league
  - Add and delete teams from leagues (promotion and relegation)
  - Track which players belong to which teams

Fundamentals

Lists

- *Ordered, mutable* collections: like a collection of numbered buckets!
- Can mutate, sort, and access different elements of lists

Lists: Initialization

- Initialize a list of player surnames:

```
barca=[‘valdes’,
’alves’,’xavi’,’iniesta’,
’messi’]
```

- Access elements (individual player surnames) by index:

```
>> print barca[0]
>> ‘valdes’
```

Lists: Iteration

- How can we print out all elements of the list, using a few lines of code?
  - Iteration over the items in the list

```
for player in barca:
 print player
```

  - Iteration over indices

```
for index in range(len(barca)):
 print barca[index]
```

- The simpler solution is usually better!

Lists: Operations

- Create new lists by ‘slicing’ existing lists:
  - Given: `example_list = [0,1,1,2,3,5]`
  - `first_three = example_list[:2]`
  - `last_four = example_list[2:]`

example_list
0
1
1
2
3
5

first_three
0
1
1

last_four
1
2
3
5

Lists: Operations

- Example: Relegating teams from and promoting teams to the Premier League
- Promote the top two from FLChamp10 (list)
- Relegate the bottom two from Premier10 (list)

Premier10
Man. Utd.
Chelsea
Man. City
Arsenal
Tottenham
Liverpool
Blackpool
West Ham

FLChamp10
Queens Park
Swansea City
Cardiff City
Reading
Nottingham Forest
Sheffield United
Scunthorpe

Premier 11

Lists: Operations

- Concatenating lists, we can assign Premier11
- `Premier11=Premier10[:5]+ FLChamp10[:2]`

Premier10[:5]
Man. Utd.
Chelsea
Man. City
Arsenal
Tottenham
Liverpool

FLChamp10[:2]
Queens Park
Swansea City

Premier11
Man. Utd.
Chelsea
Man. City
Arsenal
Tottenham
Liverpool
Queens Park
Norwich City

Lists: Operations

- Add:
  - `barca.append(`rossi')` adds `rossi' to the end of the list
  - `barca.insert(`rossi',0)` adds `rossi' at index 0 of the list (the beginning)
- Remove:
  - `barca.remove(`messi')` removes the first instance of `messi' from barca
- Sort
  - `barca.sort()` sorts all elements of the list in alphabetical order
- Pop
  - `barca.pop(k)` removes the kth element from the list and returns it.

Tuples: Introduction

- Essentially an **immutable** list
  - **CANNOT** change list items
  - Form: `tuple=('a', 'b', 'c', 'd',…)`

- **We saw an example of this earlier:**
  - `barca_tuple=('valdes', 'alves','xavi','iniesta', 'messi')`

Tuples: Manipulation

- **NOTICE:**
  - `tuple[0] = 'A'` returns an error

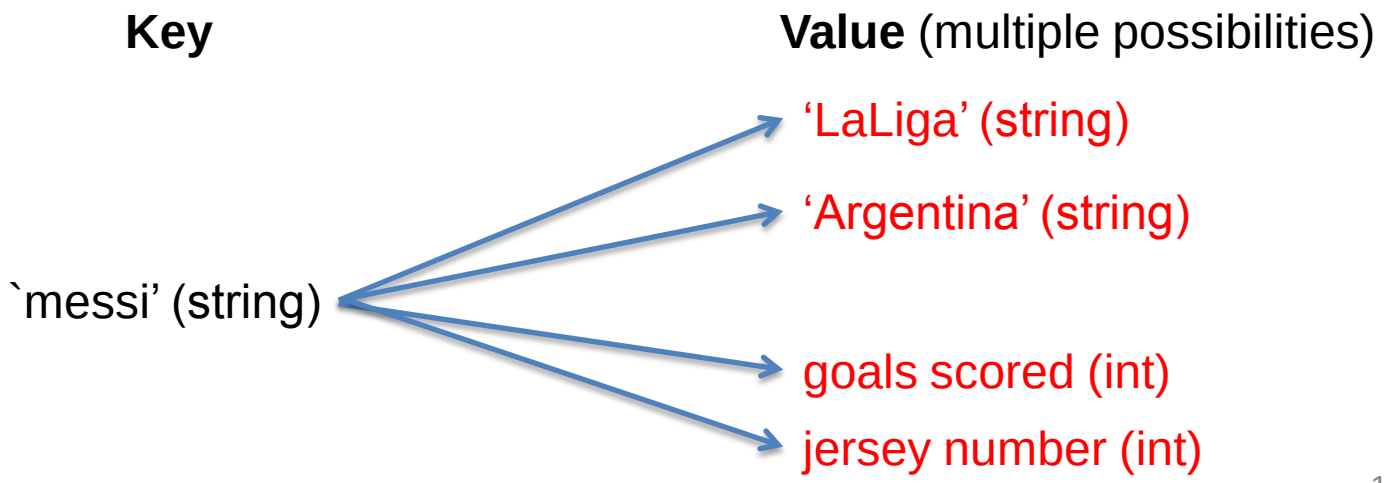
- There are some ways around this
  - Make new tuple and add part of existing tuple
  - `tuple = ('A',) + tuple[1:]`
  - New Tuple: `('A', 'b', 'c', 'd', 'e')`

Lists and Tuples: Limitations

- Suppose ~1000 players in each professional football league
- How do we check that Messi is in the league? Are there any shortcuts?
  - Sorted lists can help
  - Costly to insert new elements into sorted lists
- A different solution: **dictionaries**, a common Python implementation of **hash tables**

Dictionaries

- An unordered collection of (key,value) pairs
- (key, value) pairs are mappings
  - key: something you know
  - value: something you want to know that is related to the key
- Key and value can be objects of any type



Dictionaries: Initialization

- Initialization (maps players to teams):

```
player_team = {'messi':'barca',
'donovan':'galaxy'),
'drogba':'chelsea')}
```

Key	Value
messi	barca
donovan	galaxy
drogba	chelsea

Dictionaries: Modification

- Modification
  - Change Messi's team:
    `player_team['messi']='real_madrid'`

Key	Value
messi	real_madrid
donovan	galaxy
drogba	chelsea

Dictionaries: Modification

- Modification:
  - Add a new player:
    `player_team['beckham']='who_knows'`

Key	Value
messi	real_madrid
donovan	galaxy
drogba	chelsea
beckham	who_knows

Dictionaries

- Suppose someone gives you a list of players, `player_list`
- How can we use our dictionary, `player_team`, to print out the teams of each player on the `player_list`?
- We may not know that `player_team` has an entry for an item in `player_list`
- ```
  def check_list(player_list):   player_league = {'messi':'LaLiga',  'donovan':'MLS'}
   for item in player_list:
      if item in player_league:
         print player_league[item]
      else:
         print 'unknown league'
  ```
- Later on: exception handling
  ```

Useful Questions

- Is the data I'm storing going to change?
  - Mutability VS Immutability
  - If NOT → *Tuples!*

- If data will change? Can it fit into a single list?
  - If YES → *Use a List!*
  - Recall it has: *add*, *remove* and *sort* methods

Useful Questions

- Will one set of data be mapped to another?
  - Words to definitions, soccer players to jersey sizes, students to grades
  - Dictionary!