

NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 4, Lecture 5

Madhavan Mukund, Chennai Mathematical Institute

<http://www.cmi.ac.in/~madhavan>

Tuples

- * Simultaneous assignments

```
(age,name,primes) = (23,"Kamal",[2,3,5])
```

- * Can assign a “tuple” of values to a name

```
point = (3.5,4.8)  
date = (16,7,2013)
```

- * Extract positions, slices

```
xcoordinate = point[0]  
monthyear = date[1:]
```

- * Tuples are immutable

```
date[1] = 8 is an error
```


Generalizing lists

- * $l = [13, 46, 0, 25, 72]$
- * View l as a function, associating values to positions
 - * $l : \{0, 1, \dots, 4\} \rightarrow \text{integers}$
 - * $l(0) = 13, l(4) = 72$
- * $0, 1, \dots, 4$ are **keys**
- * $l[0], l[1], \dots, l[4]$ are corresponding **values**

Dictionaries

- * Allow keys other than `range(0,n)`
- * Key could be a string

```
test1["Dhawan"] = 84  
test1["Pujara"] = 16  
test1["Kohli"] = 200
```

- * Python **dictionary**
 - * Any immutable value can be a key
 - * Can update dictionaries in place —mutable, like lists

Dictionaries

- * Empty dictionary is {}, not []
- * Initialization: test1 = {}
- * Note: test1 = [] is empty list, test1 = () is empty tuple
- * Keys can be any immutable values
 - * int, float, bool, string, tuple
 - * But not lists, or dictionaries

Dictionaries

- ✱ Can nest dictionaries

```
score["Test1"]["Dhawan"] = 84  
score["Test1"]["Kohli"] = 200  
score["Test2"]["Dhawan"] = 27
```

- ✱ Directly assign values to a dictionary

```
score = {"Dhawan":84, "Kohli":200}  
score = {"Test1":{"Dhawan":84,  
    "Kohli":200}, "Test2":{"Dhawan":50}}
```


Operating on dictionaries

- * `d.keys()` returns sequence of keys of dictionary `d`
for `k in d.keys()`:
 # Process `d[k]`
- * `d.keys()` is not in any predictable order
for `k in sorted(d.keys())`:
 # Process `d[k]`
- * `sorted(l)` returns sorted copy of `l`, `l.sort()` sorts `l` in place
- * `d.keys()` is **not** a list — use `list(d.keys())`

Operating on dictionaries

- * Similarly, `d.values()` is sequence of values in `d`

```
total = 0
for s in test1.values():
    total = total + test1
```

- * Test for key using `in`, like list membership

```
for n in ["Dhawan", "Kohli"]:
    total[n] = 0
    for match in score.keys():
        if n in score[match].keys():
            total[n] = total[n] + score[match][n]
```


Dictionaries vs lists

- * Assigning to an unknown key inserts an entry

```
d = {}
```

```
d[0] = 7    # No problem, d == {0:7}
```

- * ... unlike a list

```
l = []
```

```
l[0] = 7    # IndexError!
```


Summary

- * Dictionaries allow a flexible association of values to keys
 - * Keys must be immutable values
- * Structure of dictionary is internally optimized for key-based lookup
 - * Use `sorted(d.keys())` to retrieve keys in predictable order
- * Extremely useful for manipulating information from text files, tables ... — use column headings as keys