Python language: Data structures and functions

The FOSSEE Group

Department of Aerospace Engineering
IIT Bombay

SciPy.in 2010, Tutorials

Outline

1. Data structures
    - Lists
    - Tuples
    - Dictionaries
    - Sets

Outline

1. Data structures
    - Lists
    - Tuples
    - Dictionaries
    - Sets

Outline

1. Data structures
   - Lists
   - Tuples
   - Dictionaries
   - Sets

Lists

We already know that

```
num = [1, 2, 3, 4]
```

is a list

Lists: methods

```
In []: num = [9, 8, 2, 3, 7]

In []: num + [4, 5, 6]
Out[]: [9, 8, 2, 3, 7, 4, 5, 6]

In []: num.append([4, 5, 6])

In []: num
Out[]: [9, 8, 2, 3, 7, [4, 5, 6]]
```

Lists: methods

```
In []: num = [9, 8, 2, 3, 7]

In []: num.extend([4, 5, 6])
In []: num
Out[]: [9, 8, 2, 3, 7, 4, 5, 6]

In []: num.reverse()
In []: num
Out[]: [6, 5, 4, 7, 3, 2, 8, 9]

In []: num.remove(6)
In []: num
```

List containership

Recall **`num`** is `[9, 8, 2, 3, 7]`

```
In []: 4 in num
Out[]: False

In []: b = 8
In []: b in num
Out[]: True

In []: b not in num
Out[]: False
```

Outline

1. Data structures
   - Lists
   - Tuples
   - Dictionaries
   - Sets

Tuples: Immutable lists

```
In []: t = (1, 2, 3, 4, 5, 6, 7, 8)

In []: t[0] + t[3] + t[-1]
Out[]: 13

In []: t[4] = 7
```

Note:

- Tuples are immutable - cannot be changed

Tuples: Immutable lists

```
In []: t = (1, 2, 3, 4, 5, 6, 7, 8)

In []: t[0] + t[3] + t[-1]
Out[]: 13

In []: t[4] = 7
```

Note:

- Tuples are immutable - cannot be changed

A classic problem

Interchange values

How to interchange values of two variables?

Note:

This Python idiom works for all types of variables.
They need not be of the same type!

A classic problem

Interchange values

How to interchange values of two variables?

Note:

This Python idiom works for all types of variables.
They need not be of the same type!

Outline

1. Data structures
    - Lists
    - Tuples
    - Dictionaries
    - Sets

Dictionaries: Introduction

- Lists index using integers
  Recall `p = [2, 3, 5, 7]` and
  `p[1]` is equal to `3`
- Dictionaries index using strings

Dictionaries . . .

```
In []: d = {'png' : 'image file',
      'txt'  : 'text file',
      'py'   : 'python code',
      'java' : 'bad code',
      'cpp'  : 'complex code'}

In []: d['txt']
Out[]: 'text file'
```

Dictionaries . . .

```
In []: 'py' in d
Out[]: True

In []: 'jpg' in d
Out[]: False
```

Dictionaries . . .

```
In []: d.keys()
Out[]: ['cpp', 'py', 'txt', 'java', 'png']

In []: d.values()
Out[]: ['complex code', 'python code',
        'text file', 'bad code',
        'image file']
```

Inserting elements into dictionary

`d[key] = value`

```
In []: d['bin'] = 'binary file'
In []: d
Out[]:
{'bin': 'binary file',
 'cpp': 'complex code',
 'java': 'bad code',
 'png': 'image file',
 'py': 'python code',
 'txt': 'text file'}
```

Duplicate keys are overwritten!

Dictionaries: containership

```
In []: 'bin' in d
Out[]: True

In []: 'hs' in d
Out[]: False
```

Note

- We can check for the containership of keys only
- Not values

Dictionaries: methods

```
In []: d.keys()
Out[]: ['bin', 'java', 'py', 'cpp', 'txt'

In []: d.values()
Out[]:
['binary file',
 'bad code',
 'python code',
 'complex code',
 'text file',
 'image file']
```

Problem Set 2.1: Problem 2.1.1

You are given date strings of the form “29 Jul, 2009”, or “4 January 2008”. In other words a number, a string and another number, with a comma sometimes separating the items.
Write a program that takes such a string as input and prints a tuple (yyyy, mm, dd) where all three elements are ints.

Outline

1. Data structures
   - Lists
   - Tuples
   - Dictionaries
   - Sets

Sets

- Simplest container, mutable
- No ordering, no duplicates
- usual suspects: union, intersection, subset . . .
- >, >=, <, <=, in, . . .

```
In []: f10 = set([1,2,3,5,8])

In []: p10 = set([2,3,5,7])

In []: f10 | p10
Out[]: set([1, 2, 3, 5, 7, 8])
```

Set . . .

```
In []: f10 & p10
Out[]: set([2, 3, 5])

In []: f10 - p10
Out[]: set([1, 8])

In []: p10 - f10, f10 ^ p10
Out[]: (set([7]), set([1, 7, 8]))

In []: set([2,3]) < p10
Out[]: True
```

Set . . .

```
In []: set([2,3]) <= p10
Out[]: True

In []: 2 in p10
Out[]: True

In []: 4 in p10
Out[]: False

In []: len(f10)
Out[]: 5
```

Problem set 2.2: Problem 2.2.1

Given a dictionary of the names of students and their marks, identify how many duplicate marks are there? and what are these?

Problem 2.2.2

Given a list of words, find all the anagrams in the list.

What did we learn?

- Advanced Data structures:
    - Lists
    - Tuples
    - Dictionaries
    - Sets