Overview

Dictionary
Functions
Modules
Files

Comments
Documentation

Dictionaries

Dictionaries allow us to collect all kinds of objects – like lists and tuples
For each item in a dictionary, there is an accompanying key
Key – value pair
D = {’key1′:’dog’,
2:’cat’,
(’key’, ‘number’, ‘three’):’mouse’}

Dictionaries can be modified in place
They are not ordered sequences – not possible to use index

```
>>>D['key1']
‘dog’
>>>D[2]
‘cat’
```

Changing the value of a key:

```
>>>D['key1'] = ['milk', 'cake']
>>>D
{(’key’, ‘number’, ‘three’): ‘mouse’, 2: ‘cat’, ‘key1′: ['milk', 'cake']}
```

Deleting from Dictionaries:

```
>>>del(D[('key', 'number', 'three')])
>>>D
{2: ‘cat’, ‘key1′: ['milk', 'cake'], ‘new’: 1.5}
```

Viewing the Keys and Values:

```
>>>D.keys()
[2, 'key1', 'new']
>>>D.values()
['cat', ['milk', 'cake'], 1.5]
```

“truth” Tests

```
>>>5 > 4
1

>>>4 == 5
0
```

Functions

set of commands grouped together as a unit with a name

Defining a function :

```
>>>def average(num1, num2):
...  avrg=(num1 + num2)/2
...  return avrg

>>>average(2,3) ← calling a function
```

Parameters can have default values

```
def personal_info(name, phone, country =
‘USA’):
```

NameSpace

Built-in
- all the core functions and constants of Python are defined

Global
- namespace at the top level of a Python module

local
- where names defined in a function reside

Nested
- the namespace of the enclosing scope

```
>>>Var = 1
>>>def func(val):
...   global Var
...   Var =val*2
...
>>>func(3)     #Calling the function
>>>print Var     #Checking the value of Var
6
```

Modules

Python modules are files with Python code
Modules are the natural way to group related code

>>>import os
>>>os.getcwd()
‘/home/me’

```
the import statement to gain access to all the
classes, functions and variables defined in a
module called os
```

Importing specific items from a module:

```
>>>from os import listdir, getcwd
>>>listdir(getcwd())
```

Making Modules :

Add the following commands into a regular text file, and name the file test.py:

```
def greet(name):
  print “Hello there, %s” %(name)
greet(’John’)
```

```
>>>import test
Hello there, John

>>>test.greet(’Pravin’)
Hello there, Pravin
```

```
import sys
  for item in sys.argv:
    print item
```

We could then run this script with a call like this:

```
/home/m/MyDocs/Test>python test.py First
test.py
First
```

Working with Files

Working with files implies being able to open, read, and write files

```
>>>f=open('MyFile.txt', 'w')
>>>f.write("testing 1... 2... 3...")
>>>f.close()

>>>f=open('MyFile.txt', 'r')
>>>FileContents=f.readlines() ← f.read(), reads entire content at once
>>>f.close()

>>>f=open('MyFile.txt', 'r')
>>>for line in f:
…       print line
>>>f.close()
```

Existing Code analysis :

```
#importing different modules from python's library.
import ConfigParser
import random
import os

def generate(easyQ, mediumQ, hardQ, nameOfFileWithQuestions):

    #reading the config file and fetching the questions from the sections present there.
    config = ConfigParser.ConfigParser()
    config.read(nameOfFileWithQuestions)
    easyQuestionSet = config.get('EasyQuestions', 'questionSet')
    mediumQuestionSet = config.get('MediumQuestions',
'questionSet')
    hardQuestionSet = config.get('HardQuestions', 'questionSet')

    #splitting the questions inside the already segregated list so that the questions can be
easily selected when required.
    easyQuestionSet = easyQuestionSet.split("::")
    mediumQuestionSet = mediumQuestionSet.split("::")
    hardQuestionSet = hardQuestionSet.split("::")
```

```
#randomly selecting the questions after splitting them.
    easyRandomSet = random.sample(xrange(len(easyQuestionSet) - 1), easyQ)
    mediumRandomSet = random.sample(xrange(len(mediumQuestionSet) - 1), mediumQ)
    hardRandomSet = random.sample(xrange(len(hardQuestionSet) - 1), hardQ)

    #storing the randomly selected questions in a list so that it can be later saved into multiple file formats.
    easyQuestionSet = [easyQuestionSet[x] for x in easyRandomSet]
    mediumQuestionSet = [mediumQuestionSet[x] for x in mediumRandomSet]
    hardQuestionSet = [hardQuestionSet[x] for x in hardRandomSet]

    #calls the function which choose the required file format in which the list of questions created is to be stored.
    print easyQuestionSet
    print mediumQuestionSet
    print hardQuestionSet
    print fileFormatToSaveIn
```

```
if __name__ == "__main__":
    generate(2,2,2, 'questionFile')
```

All functions.
Name of main function is __main__