1MD3 Tutorial 4 – Basic Data Type and PyUnit

CAO Shiqi

January 31, 2006

1 Basic Data Type

1.1 List

List is a common data structure. Usually the abstract data type of list is defined by the some operators such as create an empty list, insert an element to a list, remove an element from a list and so on. In python list is a very useful data type. To create a list, one can just write

```
>>> li = [2,1,4,2]
```

In python an index can refer an element in a list. For example,

```
>>> li[0]
1
```

Since list supports indexing there is no need of array. It is pretty nice to support negative indexing in Python.

```
>>> li[-1]
2
```

This feature is called *syntax sugar*. Syntax sugar is not necessary which means you can alway do it without this feature. However syntax sugar is handy and makes program concise. The negative indexing `li[-1]` is equivalent to `li[len(li) - 1]` where `len()` returns the length of a list. There are more sugar of list indexing, try the following by your own.

```
>>> li[1:-1]
>>> li[:3]
>>> li[-1:]
>>> li[:]
```

Searching an element in a list is also easy in Python. There is a method of list type called `index(n)`. It searches the first occurrence of `n` in a list and returns its index if it exists.

```
>>>li.index(2)
0
```

Deleting an element is to call another method `remove(n)`. It removes the first occurrence of `n` from a list.

```
>>>li
[2,1,4,2]
>>>li.remove(2)
>>>li
[1,4,2]
```

`extend(a)` is a method to add the list `a` to the end of the original list. `append(a)` is a method to add an element `a` to the end of the original list. `append(a)` is the same as `extend([a])`
Example:

```
>>> li
[3, 2, 4, 1]
>>> li
[3, 2, 4, 1]
>>> li.extend([90,100])
>>> li
[3, 2, 4, 1, 90, 100]
>>> li.append([90,100])
>>> li
[3, 2, 4, 1, 90, 100, [90, 100]]
>>> li.extend([[90,100]])
>>> li
[3, 2, 4, 1, 90, 100, [90, 100], [90, 100]]
```

1.2 Tuple

Tuple is similar to list, except once a tuple is created it can not be changed and it has no `index()`, `extend()`, `append()` methods.

```
>>> x = (li,li[3])
>>> x
([3, 2, 4, 1, 90, 100, [90, 100], [90, 100]], 1)
>>> x[0]
[1001, 2, 4, 1, 90, 100, [90, 100], [90, 100]]
>>> x[-1]
1
>>> x[0] = 2
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object does not support item assignment
```

2 PyUnit

PyUnit is a testing module in the python's library. In this framework user only needs to create testing cases, then PyUnit will run each case and print useful information.
Example:

```
import unittest
```

```
class Test(unittest.TestCase):
    def dec2bin(self,n):
        a = ""
        while n > 0:
                if n % 2 == 0:
                        a = "0" + a
                else:
                        a = "1" + a
                n = n / 2
        return a

    def tearDown(self):
"""Call after every test cases"""

    def testA(self):
self.assertEqual(self.dec2bin(0),"0")

    def testB(self):
assert self.dec2bin(8) == "1000"

    def testC(self):
assert self.dec2bin(10) == "100"

if __name__ == "__main__":
    unittest.main()
```