Python Open Labs

Collections

Acknowledgement: Kunal Baweja

What is a collection ?

What is a collection ?

- Multiple values/objects stored in a single variable
- Multiple locations, one for each value within a variable.
- lists, dictionaries, tuples, set

- Ways to locate different values within a variable.

Dictionaries

- Similar to human language dictionaries: pairs of unique keys and values
- BUT unordered

```
{
    “first”: something,
    “second”: another
}
```

Dictionaries

- Use keys to locate unique locations within dictionary
- Iterable
- Values need not be unique

```
stock = dict()

stock['pen'] = 10
stock['paper'] = 4

print stock
```

Dictionaries

- Extremely Fast and Organised
- Easy manipulation
- Costly computation

```
stock = dict()

stock['pen'] = 10
stock['paper'] = 4

print stock
stock.pop('pen')
print stock
```

Dictionaries

- keys - str or int (unique)
- Non-linear data structure (unlike lists or arrays)

```
stock = dict()

stock['pen'] = 10
stock['paper'] = 4

print stock['eraser'] ??
```

```
stock = {
          'pen': 10,
          'paper': 4
        }

print stock
print stock['eraser'] ??
```

Dictionaries: Practise

- Check if eraser exists in stock
- If eraser exists print the count of eraser else print 0

```
stock = {
            'pen': 10,
            'paper': 4,
            'pin': 100
          }

????
```

Dictionaries: Practise

- Check if eraser exists in stock
- If eraser exists print the count of eraser else print 0

```
stock = {
          'pen': 10,
          'paper': 4,
          'pin': 100
        }

if 'eraser' in stock:
    print stock['eraser']
else:
    print 0
```

Dictionaries: Practise

- KeyError is very common
- Set optional default return value with .get()

```
stock = {
          'pen': 10,
          'paper': 4,
          'pin': 100
        }

print stock.get('eraser', 0)
```

Dictionaries: More Practise

- Given a list of items construct a dictionary of their counts
- items = [‘pen’, ‘paper’, ‘eraser’, ‘pen’, ‘pen’, ‘paper’]

Tuples

- Similar to a list
- Ordered collection of python objects

```
items = (1,2,3)
items = tuple([4,5,6])
```

Tuples

- Similar to a list
- Ordered collection of python objects
- Iterable, access by index location (same as list)

```
items = (1,2,3)
print items[0]
items = tuple([4,5,6])
print items[2]
```

Tuples

- Immutable - what does that mean ?

```
items = (1,2,3)
items[0] = 15
????
```

Tuples

- Immutable - what does that mean ?
- Immutable = once a tuple is created it can’t be changed
- Can not assign, sort, delete, add, append objects etc

```
items = (1,2,3)
items[0] = 15
TypeError: 'tuple' object does not support item assignment
```

Tuples: Attention to detail

```
items = (1)


print items ????
```

```
items = (1,)


print items ????
```

Tuples

- Tuples can’t be sorted
- But a list of tuples can be
- So, given a dictionary

  use the .items() to get a

  list of (key, value) tuples
- Sort that list of tuples in

  reverse order

```
stock = {
            ‘pen’: 10,
            ‘paper’: 4,
            ‘staple’: 27
            ‘pin’: 100,
            ‘eraser’: 15
          }
```