# Session 6: Lists and Dictionaries

## 1. Basic Syntax for Python Lists

### 1.1 Indexing and slicing

```
[1]: l=[10,20,'Python',int]
     len(l)
```

```
4
```

```
[2]: l[1]
```

```
20
```

```
[3]: l[-1]
```

```
int
```

```
[4]: l[0:3]
```

```
[10, 20, 'Python']
```

```
[5]: l[1:]
```

```
[20, 'Python', int]
```

```
l[4]
```

```
IndexError: list index out of range
```

```
[6]: 10 in l
```

```
True
```

```
[7]: 50 in l
```

```
False
```

```
[8]: 50 not in l
```

```
True
```

### 1.2 Modifying lists

```
[9]: l=[10,20,'Python',int]
     l+[50,60]
```

```
[10, 20, 'Python', int, 50, 60]
```

```
[10]: l.append(40)
      l.append(80)
      l
```

```
[10, 20, 'Python', int, 40, 80]
```

```
[11]: del l[4:]
      l

[10, 20, 'Python', int]

[12]: l[2]=10
      l

[10, 20, 10, int]
```

**Q1-a:** Define a list `m=[5,4,3,8,2,[4,2]]`. Find the length of the list and the type of the last element.
**Q1-b:** Use indexing to obtain a slice of the list from 4 to 8.
**Q1-c:** Remove the last three elements of the list.
**Q1-d:** Find two ways of adding the elements 2 and 1 to `m`, using `append` and + respectively.
**Q1-e:** Modify the list so that the element 4 becomes a string `"Four"`.

### 1.3 String parsing using lists

```
[20]: s='Python for Business Analytics!!??\n'
      s.strip()

'Python for Business Analytics!!??'

[21]: s.strip().strip('?!')

'Python for Business Analytics'

[23]: s=s.strip().strip('?!')
      s

'Python for Business Analytics'

[24]: s.split(' ')

['Python', 'for', 'Business', 'Analytics']

[25]: s.split(' ')[2]

'Business'

[26]: l=s.split(' ')
      l[1:]

['for', 'Business', 'Analytics']
```

**Q2:** Given a string of the following format from the mailbox data

```
s="From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008\n"
```

**a)** Use string splitting to obtain the variables `dayOfWeek`, `month`, `day`, and `year`. In this example, `dayOfWeek='Sat'`, `month='Jan'`, `day=5` (convert to integer), `year=2008` (convert to integer).
**b)** Use string splitting to obtain the domain name of the email address. In the above example, the domain name is `'uct.ac.za'`.

### 1.3 For loops and lists

```
[30]: # Iterating through a list
      l=[0,2,4,6,8]
      for item in l:
          print(item,end=' ')
```

```
0 2 4 6 8
```

```
[31]: # Creating a list iteratively
      l=[]
      for i in range(5):
          l.append(i*2)
      l
```

```
[0, 2, 4, 6, 8]
```

**Q3:** Use a `for` loop to construct a list of the first 10 squares, including 0.

```
[32]:
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

**Q4:** Given `list1=[1,2,3,2,1,'Hi',1,'Hi']`. Initialize `list2=[]`. Use a `for` loop to iterate through `list1` and adding each item to `list2` only if the item is not already in `list2`. In the end, `list2` will only contain the unique elements in `list`

```
[33]:
```

```
[1, 2, 3, 'Hi']
```

## 2. Basic Syntax for Python Dictionaries

### 2.1 Indexing

```
[34]: d={'apple':1,'orange':2,'grape':3}
      d
```

```
{'apple': 1, 'orange': 2, 'grape': 3}
```

```
[35]: len(d)
```

```
3
```

```
[36]: d['apple']
```

```
1
```

```
d[1]
```

```
KeyError: 1
```

```
[37]: 'apple' in d
```

```
True
```

```
[38]: 'rice' not in d
```

```
True
```

```
[39]: 3 in d
```

```
False
```

```
[40]: 3 in d.values()
```

```
True
```

## 2.2 Modifying

```
[2]: d={'apple':1,'orange':2,'grape':3}
     d
```

```
{'apple': 1, 'orange': 2, 'grape': 3}
```

```
[3]: d['carrots']=4
     d
```

```
{'apple': 1, 'orange': 2, 'grape': 3, 'carrots': 4}
```

```
[4]: d['apple']=5
     d
```

```
{'apple': 5, 'orange': 2, 'grape': 3, 'carrots': 4}
```

```
[5]: d['apple']+=1
     d
```

```
{'apple': 6, 'orange': 2, 'grape': 3, 'carrots': 4}
```

```
d['rice']+=1
```

```
KeyError: 'rice'
```

```
[6]: if 'rice' not in d:
         d['rice']=0
     d['rice']+=1
     d
```

```
{'apple': 6, 'orange': 2, 'grape': 3, 'carrots': 4, 'rice': 1}
```

```
[8]: if 'rice' not in d:
         d['rice']=0
     d['rice']+=1
     d
```

```
{'apple': 6, 'orange': 2, 'grape': 3, 'carrots': 4, 'rice': 3}
```

**Q5-a:** Create a dictionary `name` mapping the numbers `0` to `3` to their English name. (i.e. the key `0` should map to the value `'zero'`, and `1` should map to `'one'`, etc.)

**Q5-b:** Find the number of elements in the dictionary using `len`.

**Q5-c:** Check if 'two' is a key is in the dictionary. Check if it is a value.

**Q6:** Write a function `histogram` with taking a list named `items` as input. The function should return a dictionary mapping each unique element in the list to the number of occurences. (Hint: use a `for` loop to go through each `item` in the list and maintain a dictionary mapping each `item` to the number of occurrences so far. You can use the above code to do the incrementing.)

```
[52]: histogram([3,2,1,2,2,2])
```

```
{3: 1, 2: 4, 1: 1}
```

## 2.3 For loops and dictionaries

```
[53]: # Iterating through a dictionary
      d={'apple':5,'rice':4,'broccoli':8}
      for key in d:
          value=d[key]
          print(key,value)
```

```
apple 5
rice 4
broccoli 8
```

```
[54]: # Building a dictionary iteratively
      l=['apple','rice','broccoli']
      d={}
      for item in l:
          d[item]=len(item)
      d
```

```
{'apple': 5, 'rice': 4, 'broccoli': 8}
```

**(optional) Q7:** Write the function `squares` that takes one input argument `n` and returns a dictionary mapping each number from 0 to $n-1$ to its square.

```
[56]: squares(5)
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

## Case 7a: Mail Log Analysis I

Write a program to read through a specified mail log, filtering only for lines that start with `"From:"`. This will yield the lines as follows:

```
From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
From: zqian@umich.edu
...
```

Instead of printing the above lines, obtain the domain name (which is the string after `"@"`) of each line, and maintain a list of unique domain names. The program should print an alphabetically sorted version of this list, with one entry on each line. The following output is from the file `"mbox-short.txt"`.

```
caret.cam.ac.uk
gmail.com
iupui.edu
media.berkeley.edu
uct.ac.za
umich.edu
```

### Case 7b: Mail Log Analysis II

Instead of using a list to keep track of the unique domain names, use a dictonary to count how many messages have come from each domain name. The output of the program is similar to the above, except that you should also show how many messages come from each domain name.

```
caret.cam.ac.uk 1
gmail.com 1
iupui.edu 8
media.berkeley.edu 4
uct.ac.za 6
umich.edu 7
```

### Case 7c: Mail Log Analysis III

Analogous to the code in 7b, this time you will filter only for emails that come from the domain `"umich.edu"` and count how many messages come from each address.

```
gsilver@umich.edu 3
zqian@umich.edu 4
```