Master of Technology in Knowledge Engineering

KE5106
Semester II 2018

Lecture 6b: Web Scraping 1

Charles Pang
Institute of Systems Science
National University of Singapore
Email: charlespang@nus.edu.sg

Contents

1.Sourcing Data from the Internet

- The Internet is a huge repository of data
  - Numerical, Text, Video, Image, Voice, etc.
- For downloading small data sets:
  - Ctrl-c Ctrl-v
  - MS Excel
  - Application Programming Interfaces (APIs)
- For downloading large & diverse data sets:
  - Crawling and Scraping off webpages

Scraping the Web

- In order to scrape the web, we need to first have a basic understanding of the language used to write web pages
- Web pages are written in HTML (HyperText Markup Language)
- HTML has gone through different versions as new features have been added. The current version is HTML5.
- HTML's custodian is the World Wide Web Consortium or W3C. You can find the HTML5 specification online
- This is not intended to be a course on HTML – but an overview /refresher to get going with scraping data
- This is a good reference for HTLM: https://www.w3schools.com/

2.HTML Basics

- What is HTML?
- HTML Structure
- Tags and Attributes
- Absolute & Relative Paths
- Lists
- Scripts

What is HTML?

- HTML is a language that tells a browser how to display a web page
- The html document itself is a plain text file composed of elements called **tags**.
- Tags are enclosed in angle brackets:

  <tagname> some content </tagname>

- Tags normally come in pairs like <b> and </b>
- Tags are not case sensitive, <b> is the same as <B>
- Tags are usually embedded within other tags
- Tags are the most important concept when we do scraping

Common Tags

<head>	Header
<div>	Division
<span>	Grouping
<p>	Paragraph
<a>	Anchor
<img>	Image
<ol>	Ordered list
<ul>	Unordered list
<li>	List item
<table>	Table
<td>	Table Data
<tr>	Table Row
<script>	Client-side script

Basic HTML Structure

```
<!DOCTYPE html>
<html>

<head>
       <title> Page Title </title>
</head>

<body>

       <h1> My First Heading </h1>
       <p> My first paragraph.</p>

</body>

</html>
```

Metadata

Bodydata

Tags & Attributes

- Tags can have **attributes**
- Attributes provide **additional information** about elements
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: **name="value"**

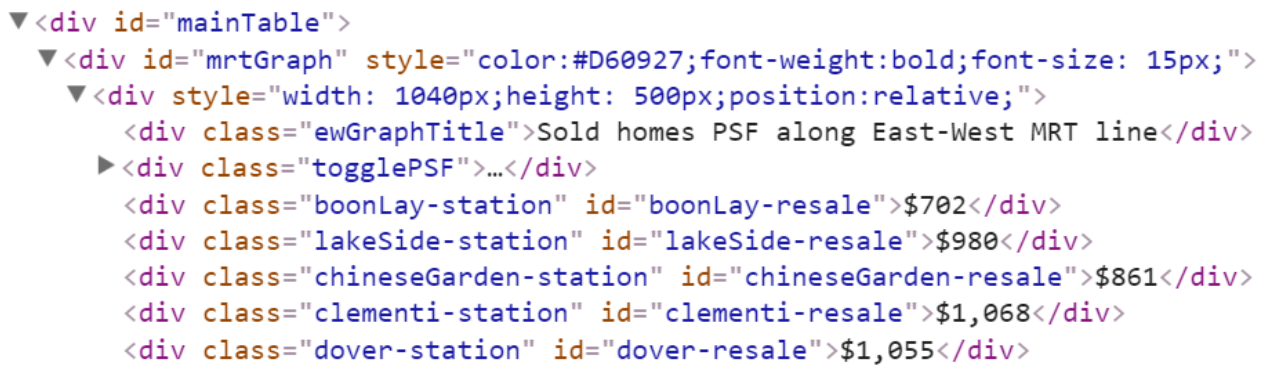
```
<div class="c-product-card__price">
    <span class="c-product-card__price-final">
        SGD 198.00
    </span>
    <span class="c-product-card__discount">-40%</span>
    <div class="c-product-card__old-price">SGD 329.00</div>
</div>
```

Block and Inline elements

- HTML elements can be divided into 2 categories :
- A block-level element occupies the entire space of its parent element (container), thereby creating a "block“ It always starts on a new line:
  - <**div**>
  - <**h1**> - <**h6**>
  - <**p**>
  - <**form**>
- An **inline** element does not start on a new line and create shorter structures compared to block-level elements. Examples are:
  - <**span**>
  - <**a**>
  - <**href**>

<div>

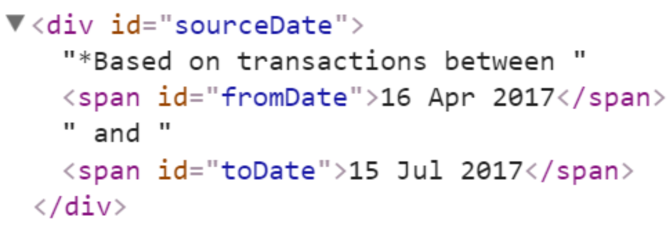
- The **<div>** element is often used as a container to group content and other HTML elements.
- <div> usually contains **class** attribute.
- When used together with CSS, the <div> element can be used to style blocks of content



```
▼<div id="mainTable">
  ▼<div id="mrtGraph" style="color:#D60927;font-weight:bold;font-size: 15px;">
    ▼<div style="width: 1040px;height: 500px;position:relative;">
        <div class="ewGraphTitle">Sold homes PSF along East-West MRT line</div>
      ▶<div class="togglePSF">…</div>
        <div class="boonLay-station" id="boonLay-resale">$702</div>
        <div class="lakeSide-station" id="lakeSide-resale">$980</div>
        <div class="chineseGarden-station" id="chineseGarden-resale">$861</div>
        <div class="clementi-station" id="clementi-resale">$1,068</div>
        <div class="dover-station" id="dover-resale">$1,055</div>
```

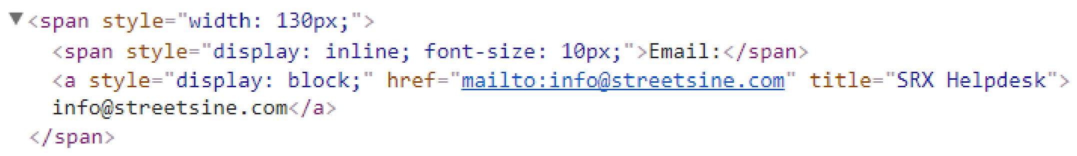
<span>

- The **<span>** inline element is often used as a container for some text.
- <span> element has no required attributes, but both **style** and **class** are common.



```
▼<div id="sourceDate">
   "*Based on transactions between "
   <span id="fromDate">16 Apr 2017</span>
   " and "
   <span id="toDate">15 Jul 2017</span>
 </div>
```

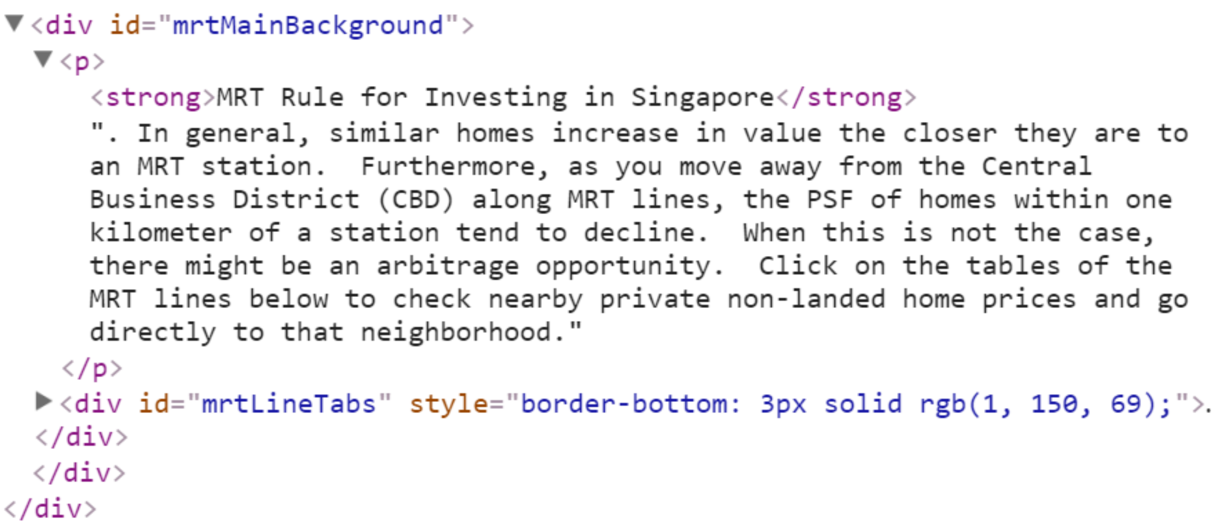
- When used together with CSS, the <span> element can be used to style parts of the text:



```
▼<span style="width: 130px;">
   <span style="display: inline; font-size: 10px;">Email:</span>
   <a style="display: block;" href="mailto:info@streetsine.com" title="SRX Helpdesk">
   info@streetsine.com</a>
 </span>
```

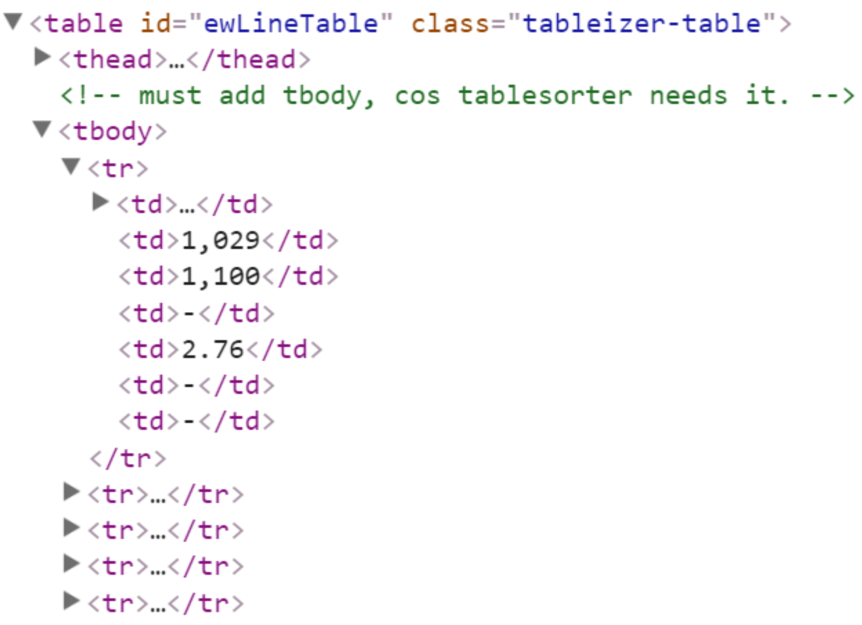
<p>

- The **<p>** element defines a **paragraph**:
- <**br**> is used for a line break (a new line) without starting a new paragraph:

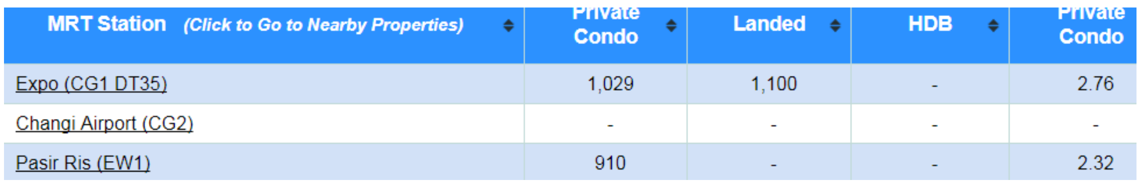


```
▼<div id="mrtMainBackground">
  ▼<p>
     <strong>MRT Rule for Investing in Singapore</strong>
     ". In general, similar homes increase in value the closer they are to
     an MRT station.  Furthermore, as you move away from the Central
     Business District (CBD) along MRT lines, the PSF of homes within one
     kilometer of a station tend to decline.  When this is not the case,
     there might be an arbitrage opportunity.  Click on the tables of the
     MRT lines below to check nearby private non-landed home prices and go
     directly to that neighborhood."
   </p>
  ▶<div id="mrtLineTabs" style="border-bottom: 3px solid rgb(1, 150, 69);">…
  </div>
  </div>
</div>
```

<table> <td> <tr>



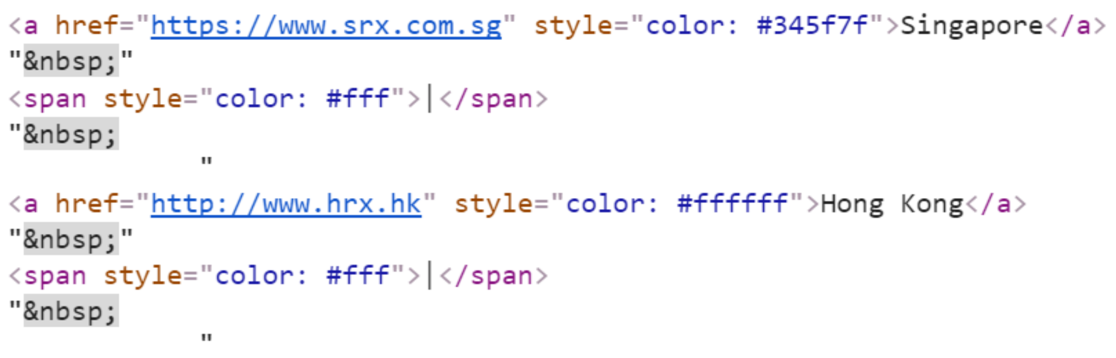
```
▼<table id="ewLineTable" class="tableizer-table">
  ▶<thead>…</thead>
   <!-- must add tbody, cos tablesorter needs it. -->
  ▼<tbody>
    ▼<tr>
      ▶<td>…</td>
       <td>1,029</td>
       <td>1,100</td>
       <td>-</td>
       <td>2.76</td>
       <td>-</td>
       <td>-</td>
     </tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
    ▶<tr>…</tr>
```



MRT Station (Click to Go to Nearby Properties)	Private Condo	Landed	HDB	Private Condo
Expo (CG1 DT35)	1,029	1,100	-	2.76
Changi Airport (CG2)	-	-	-	-
Pasir Ris (EW1)	910	-	-	2.32

<a>

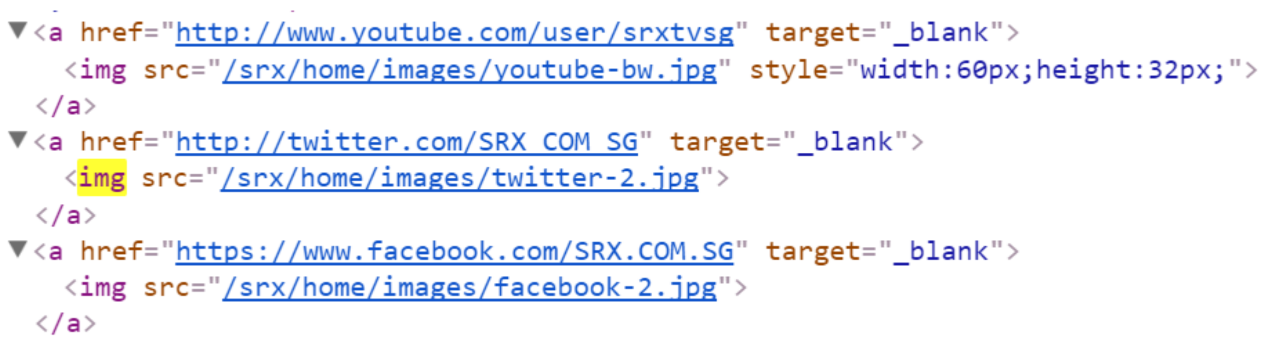
- The **<a>** tag element creates a hyperlink to other web pages, files, locations within the same page, email addresses, or any other URL (mozilla.org)
- The **href** attribute contains a URL that the hyperlink points to



```
<a href="https://www.srx.com.sg" style="color: #345f7f">Singapore</a>
"&nbsp;"
<span style="color: #fff">|</span>
"&nbsp;
           "
<a href="http://www.hrx.hk" style="color: #ffffff">Hong Kong</a>
"&nbsp;"
<span style="color: #fff">|</span>
"&nbsp;
           "
```

<img>

- <img> is used to define Images
- The **src** attribute refers to the image location
- The **alt** attribute is displayed if the image is missing,



```
▼<a href="http://www.youtube.com/user/srxtvsg" target="_blank">
   <img src="/srx/home/images/youtube-bw.jpg" style="width:60px;height:32px;">
 </a>
▼<a href="http://twitter.com/SRX_COM_SG" target="_blank">
   <img src="/srx/home/images/twitter-2.jpg">
 </a>
▼<a href="https://www.facebook.com/SRX.COM.SG" target="_blank">
   <img src="/srx/home/images/facebook-2.jpg">
 </a>
```

Absolute and Relative paths

- Points to a file located at that location:

```
<img src="https://www.w3schools.com/images/picture.jpg" alt="Mountain">
```

- Points to a file in the images folder located at the **root** of the current web:

```
<img src="/images/picture.jpg" alt="Mountain">
```

- Points to a file in the images folder located in the **current** folder:

```
<img src="images/picture.jpg" alt="Mountain">
```

- Points to a file in the images folder located in the folder one level above the current folder:

```
<img src="../images/picture.jpg" alt="Mountain">
```

Lists

<ul> defines an Unordered list

<li> defines a list element

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

- Coffee
- Tea
- Milk

<ol> defined an Ordered list

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

1. Coffee
2. Tea
3. Milk

<script>

- The **<script>** element is used to embed or reference executable code using the **src** attribute.
- Scripts makes HTML pages more dynamic and interactive. For example, a script could generate a **pop-up** alert box message, or provide a **dropdown** menu.

```
<head>
<title>Internal Script</title>
</head>
<body>
<script type="text/javascript">
   document.write("Hello Javascript!")
</script>
</body>

<head>
<script src="yourfile.js" type="text/javascript" />
</head>
```

<script>

- document.getElementById(id) method is used to select an element – usually to modify it.
- This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo“:

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello
JavaScript!";
</script>

</body>
</html>
```

3.Python Basics

- What is Python?
- Python Data Types
- Control Structures
- Functions

What is Python?

- Small and versatile programming language
- Interpretive run-time environment
- Supports multiple programming paradigms – object-oriented, imperative and functional
- Design philosophy emphasizes code readability – has clear syntax
- Has a comprehensive library for math, science
- latent (dynamic) type system

Python Background

- Complete high-level programming language named after BBC TV series Monty Python’s Flying Circus
- Implemented in 1989 by Guido van Rossum (Dutch)
- Worked at Google from 2005 – 2012, now with Dropbox.
- Voted Most Popular Programming Language of the Year for 2011-2016 (5 years in a row) by CodeEval
- Huge and thriving community of developers
- World-Class Software Companies That Use Python
  - Google, Facebook, Instagram, Dropbox, Quora, Netflix, BitTorrent, Spotify, Reddit, Yahoo Map

Python Data Types

- Strings

```
>>> name =  'Charles Pang'
>>> address = '25 Heng Mui Keng Terrace'
>>> name
'Charles Pang'
>>> print(name)
Charles Pang
```

- Integers

```
>>> age = 37
>>> salary = 18546
>>> annual = salary * 12
>>> annual
222552
>>> print("Income Tax = ", annual*0.2)
Income Tax =  44510.4
```

- Floats

```
>>> weight = 65.68
>>> height = 1.695
>>> print("Ht:",height," Wt:",weight," BMI:",weight/(height**2))
Ht: 1.695  Wt: 65.68  BMI: 22.86092185066263
```

Python Data Types (cont'd)

- Lists

```
>>> list_1 = [1,2,3,4,5]
>>> list_2 = ['N','U','S','M','T','e','c','h']
>>> list_3 = ['K','E',2,8]
>>> print(list_3)
['K', 'E', 2, 8]
>>> list_1[2]
3
>>> list_1[2] = 'X'
>>> list_1
[1, 2, 'X', 4, 5]
>>>
>>> list_4 = [[1,2,3,4],['N','U','S'],list_3]
>>> list_4
[[1, 2, 3, 4], ['N', 'U', 'S'], ['K', 'E', 2, 8]]
>>> list_4[0] = ['I','S','S']
>>> list_4
[['I', 'S', 'S'], ['N', 'U', 'S'], ['K', 'E', 2, 8]]
```

Can have mixed data types

Index starts from ZERO

Can have nested list

Operations on Lists

```
>>> list1 = [89, 34, 23, 'cat', 35, 'dog']
>>> len(list1)
6
>>> list1.append('hamster')
>>> list1
[89, 34, 23, 'cat', 35, 'dog', 'hamster']
>>> list1.reverse()
>>> list1
['hamster', 'dog', 35, 'cat', 23, 34, 89]
>>> list1.remove(34)
>>> list1
['hamster', 'dog', 35, 'cat', 23, 89]
>>> list1[3]='new'
>>> list1
['hamster', 'dog', 35, 'new', 23, 89]
>>> for item in list1:
        print(item)


hamster
dog
35
new
23
89
```

append

reverse

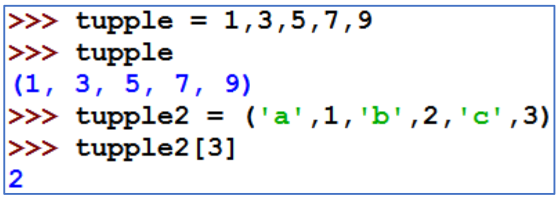
remove

replace

Iterating through a list

Python Data Types (cont'd)

- Tuples



```
>>> tupple = 1,3,5,7,9
>>> tupple
(1, 3, 5, 7, 9)
>>> tupple2 = ('a',1,'b',2,'c',3)
>>> tupple2[3]
2
```

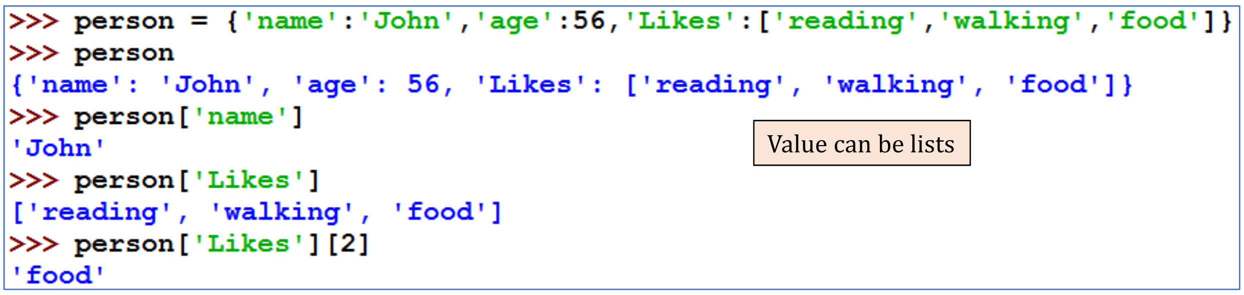
Round parantheses

Access like lists

Unlike lists, it cannot be changed: tupple[2] = 9 is illegal!

- Dictionaries

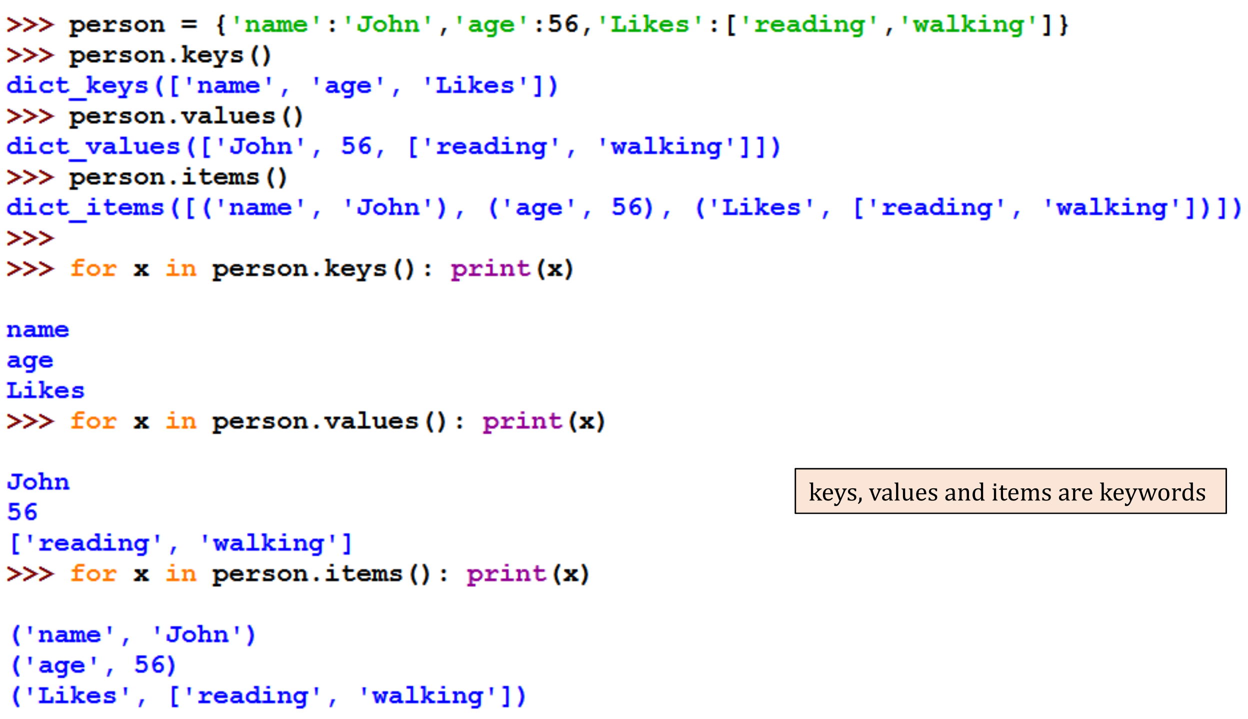
attribute:value pairs



```
>>> person = {'name':'John','age':56,'Likes':['reading','walking','food']}
>>> person
{'name': 'John', 'age': 56, 'Likes': ['reading', 'walking', 'food']}
>>> person['name']
'John'
>>> person['Likes']
['reading', 'walking', 'food']
>>> person['Likes'][2]
'food'
```

Value can be lists

Operations on Dictionaries



```
>>> person = {'name':'John','age':56,'Likes':['reading','walking']}
>>> person.keys()
dict_keys(['name', 'age', 'Likes'])
>>> person.values()
dict_values(['John', 56, ['reading', 'walking']])
>>> person.items()
dict_items([('name', 'John'), ('age', 56), ('Likes', ['reading', 'walking'])])
>>>
>>> for x in person.keys(): print(x)

name
age
Likes
>>> for x in person.values(): print(x)

John
56
['reading', 'walking']
>>> for x in person.items(): print(x)

('name', 'John')
('age', 56)
('Likes', ['reading', 'walking'])
```

keys, values and items are keywords

Iterating through "lists"

Tupples

```
>>> names=('matt','mark','luke','john')
>>> for name in names: print(name)

matt
mark
luke
john
```

Dictionary

```
>>> person = {'name':'John','age':56,'Likes':['reading','walking','food']}
>>> for key in person:
        print(key)
name
age
Likes
>>> for key in person:
        print(key,':',person[key])
name : John
age : 56
Likes : ['reading', 'walking', 'food']
```

String

```
>>> for letter in "The Quick Brown Fox ...":
        print(letter,end='')
The Quick Brown Fox ...
```

Suppress the carriage return

Control Structures – For-Loops & IF-ELSE

```
>>> x= 5
>>> range(5)
range(0, 5)
>>> for i in range(x):
        print(i)

0
1
2
3
4
>>> for i in range(x):
        if i == 3: continue
        print(i)

0
1
2
4
>>> for i in range(x):
        if i > 2: break
        print(i)

0
1
2
```

range, continue, break

```
>>> temperature = 38
>>> if temperature > 37.5:
        print("You have a fever \n")
    else:
        print("You are normal \n")
You have a fever
>>> temperature = 35.9
>>> if temperature > 37.5:
        print("You have a fever \n")
    elif temperature > 36.0:
        print("You are little hot")
    else:
        print("you are fine")
you are fine
```

If, else, elif

```
>>> a = 45
>>> print('< 45' if a <45 else '> 45')
> 45
>>> a = 99
>>> print('< 45' if a <45 else '> 45')
> 45
```

Conditional print statement

Control Structure - While Loops

```
count = 0
while count < 5:
   print (count, " is  less than 5")
   count = count + 1
else:
   print (count, " is not less than 5")
```

```
0  is  less than 5
1  is  less than 5
2  is  less than 5
3  is  less than 5
4  is  less than 5
5  is not less than 5
```

```
i = 0
numlist = []

while i < 6:
    numlist.append(i)
    i = i + 1
    print("i= %d, List = " %i, numlist)

print("The numbers: ")
for num in numlist:
    print(num)
```

```
i= 1, List =  [0]
i= 2, List =  [0, 1]
i= 3, List =  [0, 1, 2]
i= 4, List =  [0, 1, 2, 3]
i= 5, List =  [0, 1, 2, 3, 4]
i= 6, List =  [0, 1, 2, 3, 4, 5]
The numbers:
0
1
2
3
4
5
```

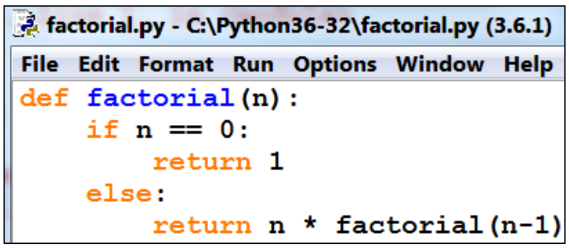
Function

- Sometimes it is useful to encapsulate a chunk of code into a black-box that can be re-used. The python functions does that.

```
>>> def half(x):
        x = x /2.0
        return(x)

>>> half(30)
15.0
>>> half(33)
16.5
```

Larger functions can be stored separately


factorial.py - C:\Python36-32\factorial.py (3.6.1)
File Edit Format Run Options Window Help

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

Default values

```
>>> def bmi(ht=1.7,wt=60):
        return(wt/(ht*ht))

>>> bmi()
20.761245674740486
>>> bmi(1.84, 75)
22.152646502835537
```

Importing the function

```
>>> import factorial as f
>>> f.factorial(12)
479001600
>>> f.factorial(5)
120
```

4.Scraping with Python

- Install Python: v3.6.2 (Win & Mac)
- Common Libraries (pip3 install …)
- Scraping Demo

Requests

- **Requests** is a Python library that allows you to send HTTP/HTTPS request and download a web page.

```
>>> import requests
>>> requests.get("https://www.iss.nus.edu.sg/")
<Response [200]>
```

- Returns a response object. The content of this object is a raw HTML file

BeautifulSoup

- **BeautifulSoup** is a Python library for parsing raw HTML and XML files. Parsing creates a parse-tree which you to be able to efficiently extract the text/data values in an HTML document
  - e.g. “Hello World” from the HTML markup:

    <p>Hello World</p>

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(html, 'html.parser')
```

- Returns a parse tree.

5.Workshop

- As per instruction.