Python - Advance Level

Lesson 3 : Web Scraping

Web Scraping

Web Scraping is a technique employed to extract large amount of data from websites whereby the data is extracted and saved to a local file in your computer or to a database.

Webpages

Web Scraping

Use Cases

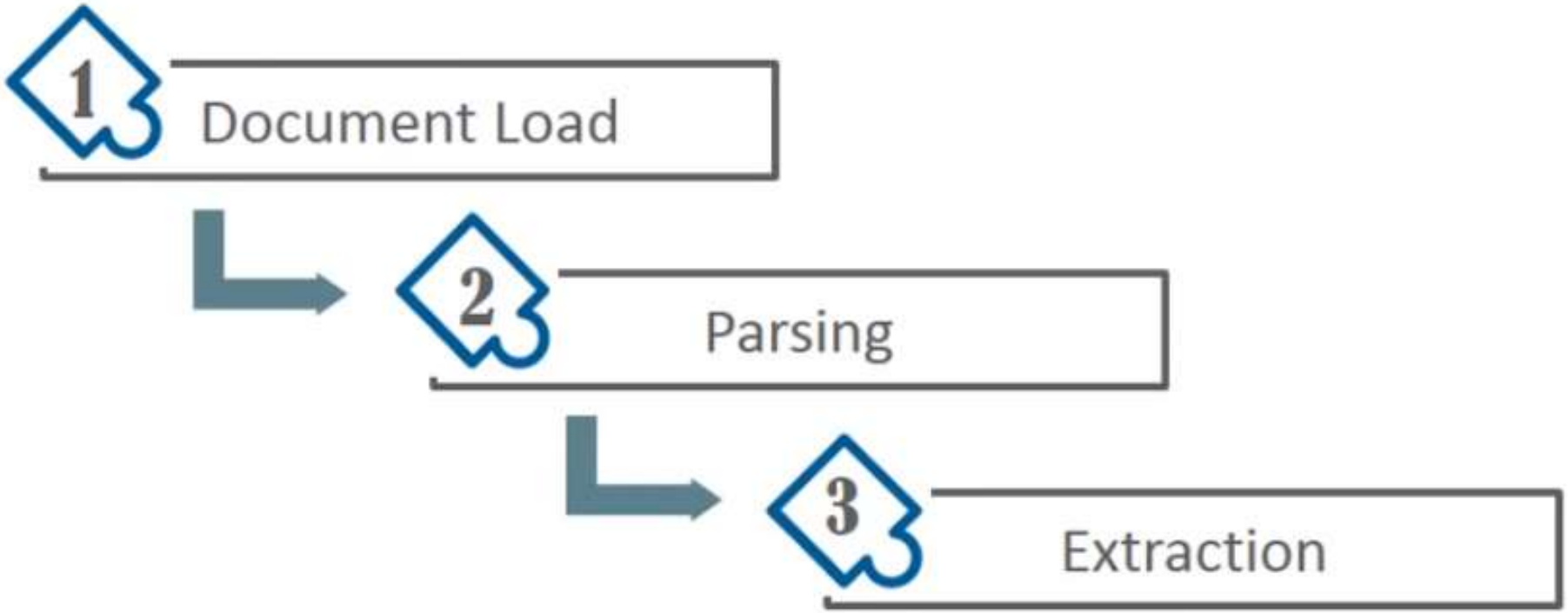
E-commerce portals

Market Research

Social Websites

Travel Websites

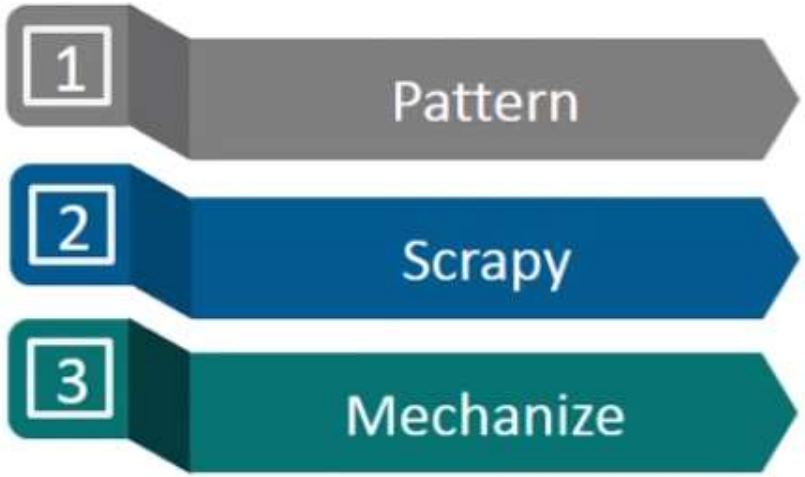
Web Scraping



Web Scraping Consideration

It’s important to read and understand the legal information and terms and conditions mentioned in the website.

Different Python Packages



Web Scraping Tool: BeautifulSoup

Some of the reasons to choose BeautifulSoup are as follows:

Efficient tool for dissecting documents and extracting information from the web pages

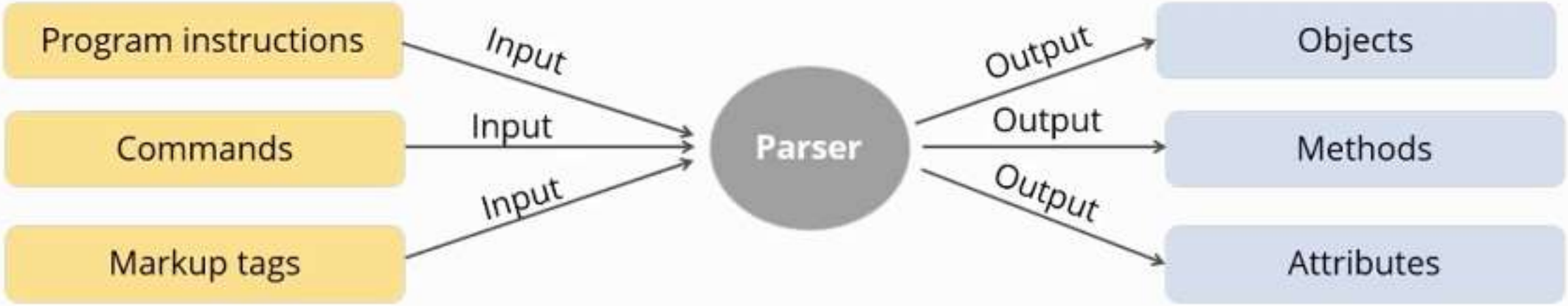
Powerful sets of built-in methods for navigating, searching, and modifying a parse tree

Possess parser that supports both html and xml documents

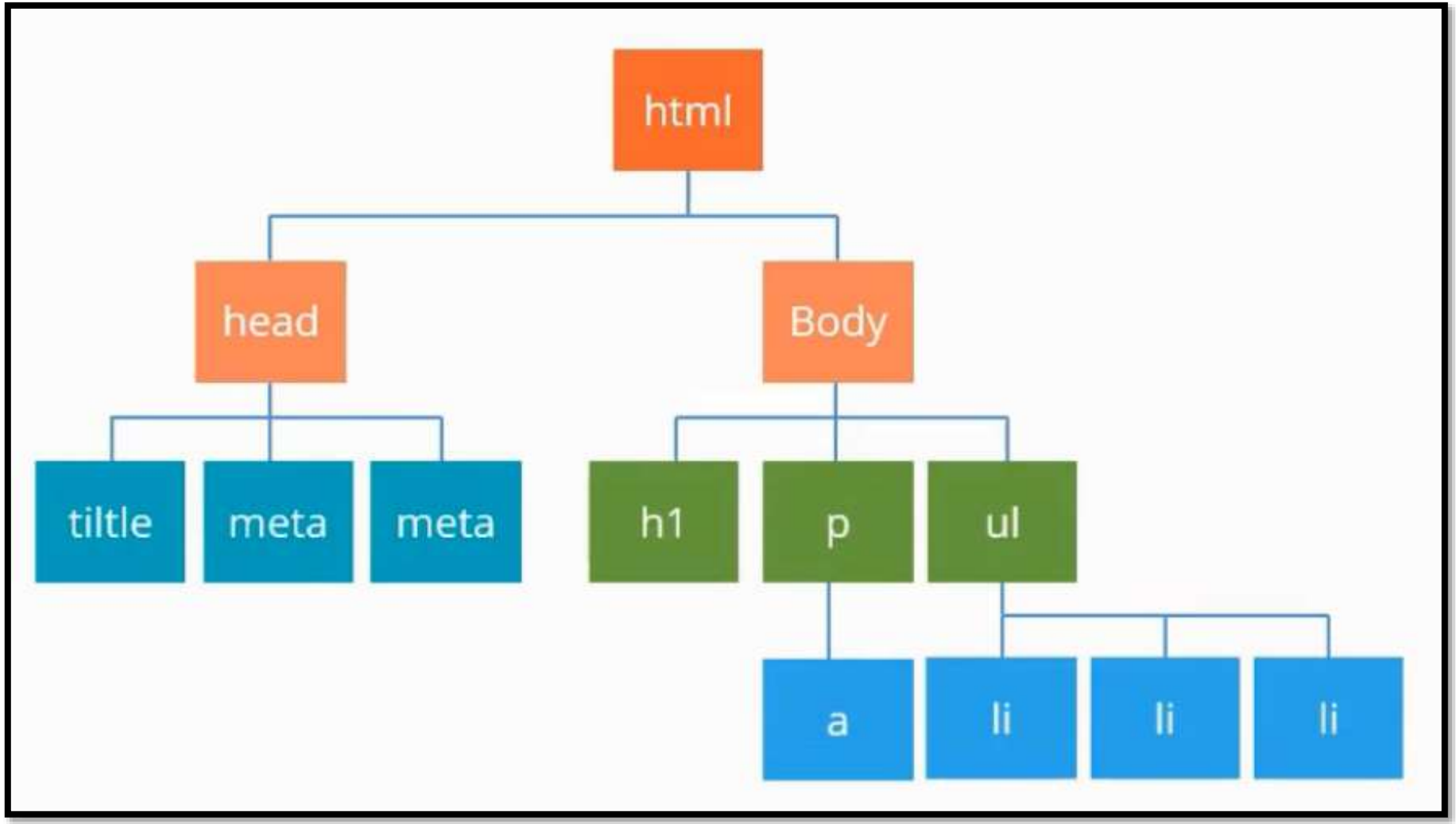
Parser

A Parser is a basic tool to interpret or render information from a web document.

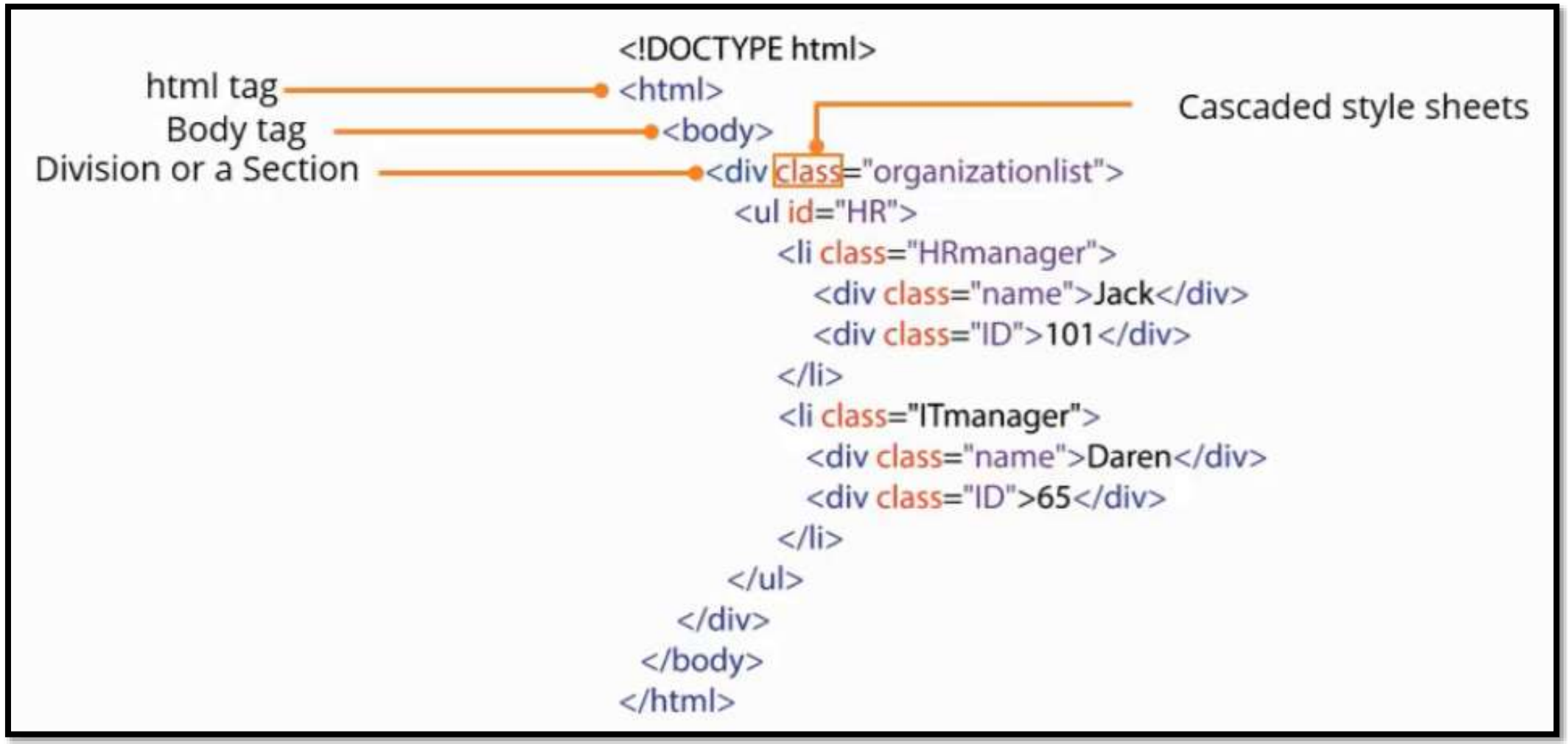
A Parser is also used to validate the input information before processing it.



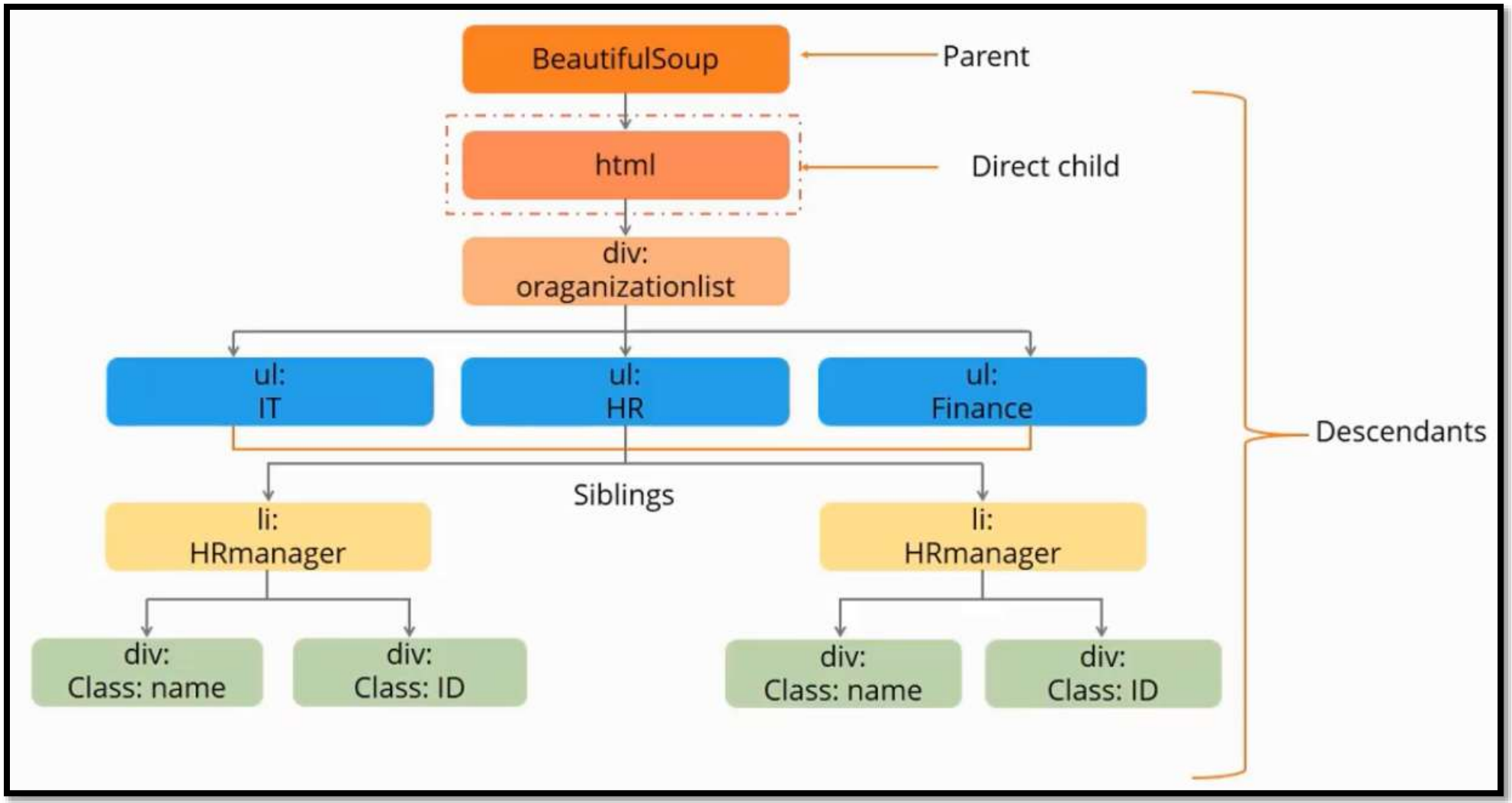
Understanding The Tree



Understanding The Tree



Understanding The Tree



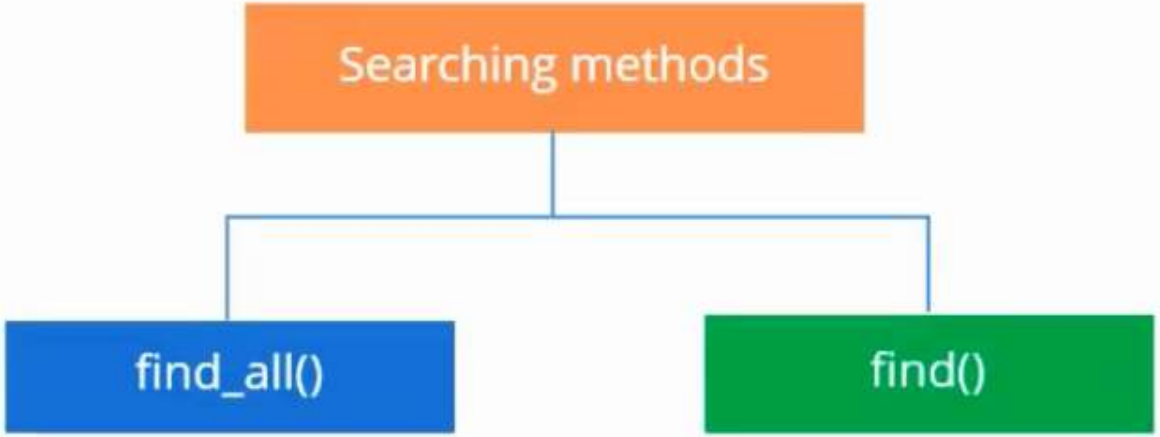
Searching The Tree - Filters

With the help of the search filters technique, you can extract specific information from the parsed document.

The filters can be treated as search criteria for extracting the information based on the elements present in the document.

Searching the Tree – find_all()

BeautifulSoup defines a lot of methods for searching the parsed tree.



Searching the tree with find()

The find_all() finds the entire document looking for results. To find one result, use find(). The find() method has a syntax similar to that of the find_all() method; however, there are some key differences.

Method name	Search Scope	Match Found	Match Not Found
Find_all()	Scans entire document	Returns list with values	Returns empty list
Find()	Searches only for passed argument	Returns only the first match value	Returns None

Re (Regular expression operations) module

Slicing Example:

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  5 2014, 20:15:28)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> string = "Ubiquitous"
>>> print(string[0:3])
Ubi
>>> string[0:4]
'Ubiq'
>>> string[0:6]
'Ubiqui'
>>> string[3:0]
''
>>> string[3:]
'quitous'
>>> string[3:5]
'qu'
>>> string[1:]
'biquitous'
>>> import re
>>> dir(re)
['A', 'ASCII', 'DEBUG', 'DOTALL', 'I', 'IGNORECASE', 'L', 'LOCALE', 'M', 'MULTILINE', 'S', 'Scanner', 'T', 'TEMPLATE', 'U', 'UNICODE', 'VERBOSE', 'X', '_MAXCA
__spec__', '__version__', '_alphanum_bytes', '_alphanum_str', '_cache', '_cache_repl', '_compile', '_compile_repl', '_expand', '_pattern_type', '_pickle', '_subx',
ompile', 'sre_parse', 'sub', 'subn', 'sys', 'template']
>>> string = "The night was cold and dark"
>>> re.search("night",string)
<_sre.SRE_Match object; span=(4, 9), match='night'>
>>>
```

Re module

Example 1:

```
<_sre.SRE_Match object; span=(4, 9), match='night'>
>>> m = re.search("night", string)
>>> print(m)
<_sre.SRE_Match object; span=(4, 9), match='night'>
>>> start = m.start()
>>> end = start + 5
>>> print(start)
4
>>> print(end)
9
>>> string[start:end]
'night'
>>>
>>> print(m.end())
9
```

Example 2:

```
>>> string2 = "asdlsakdfhkjsadfhwkjadhfas Cheese caskdjalksdj aksdj Cakejakshdkasjhd akjdh Pizza"
>>> re.search("Cheese", string2)
<_sre.SRE_Match object; span=(27, 33), match='Cheese'>
>>> position = re.search("Cheese", string2)
>>> position.start()
27
>>> position.end()
33
>>> string2[position.start() : position.end()]
'Cheese'
>>>
```

Re module

```
import re
import urllib.request
url='https://www.weather-forecast.com/locations/'
Stock=input("Enter your stock:")
url=url+Stock
print(url)
data=urllib.request.urlopen(url).read()
data1=data.decode('utf-8')
'''
print(data1)
'''
m=re.search("phrase",data1)


start=m.end()+2
'''
print(start)
'''
n=re.search(".</span></p></td><td ",data1)

end=n.start()
'''
print(end)
'''
print(data1[start:end])
```

```
==============
>>>
Enter your stock:ajman
https://www.weather-forecast.com/locations/ajman
Mostly dry. Warm (max 33&deg;C on Sat morning, min 28&deg;C on
Mon night). Winds decreasing (fresh winds from the N on Sat aft
ernoon, light winds from the ENE by Mon morning)
>>> ================================ RESTART ================
==============
>>>
Enter your stock:london
https://www.weather-forecast.com/locations/london
Moderate rain (total 10mm), heaviest on Mon night. Mild tempera
tures (max 10&deg;C on Sun morning, min 5&deg;C on Sat morning)
. Mainly strong winds
>>>
```

Exercise

Extract a lists of items from the below URL and save the item list in a text file.

https://www.emaxme.com/s001/catalogsearch/result/?category=&q=iphone

BeautifulSoup

```
from bs4 import BeautifulSoup as soup
import requests
my_url=https://www.emaxme.com/s001/catalogsearch/result/?category=&q=iphone
source=requests.get(my_url).text
page_soup=soup(source,"html.parser")
a=page_soup.select('.product-item-link')
l=len(a)

for i in range(l):
    #print(a[i].text)
    b=a[i].text
    c=str(b)
    print(c[34:])
```

Quiz 1

Which of the following libraries is used to extract a web page?

- Beautiful Soup
- Pandas
- Requests
- Numpy

Python - Advance Level

Lesson 4 : Error Handling

What is Exception Handling

Let us begin!

What is an exception?

Definition of Exception

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions

What is exception handling?

Exception Handling

Process of responding to the occurrence, during computation, of exceptional conditions requiring special processing – often changing the normal flow of program execution

Common exceptions:

ImportError: an import fails

IndexError: a list is indexed with an out-of-range number

NameError: an unknown variable is used

SyntaxError: the code can not be parsed properly

TypeError: a function is called on a value of an inappropriate type.

ValueError: a function is called on a value of the correct type, but with an inappropriate value.

Process of Exception Handling

Important Terms

try	Keyword used to keep the code segment under check
except	Segment to handle the exception after catching it
else	Run this when no exceptions exist
finally	No matter what run this code if/if not for exception

Process of Exception Handling

If an error is encountered, a **try** block code execution is stopped and transferred down to the **except** block.

In addition to using an except block after the try block, you can also use the **finally** block.

The code in the finally block will be executed regardless of whether an exception occurs

Process of Exception Handling

Visually it looks like this!

```
try:
    Run this code
except:
    Execute this code when there is an exception
else:
    No exceptions? Run this code.
finally:
    Always run this code.
```

Exception Handling In Python

Coding In Python

Coding In Python

Python code for Exception Handling

```
>>> print( 0 / 0 ))
  File "<stdin>", line 1
    print( 0 / 0 ))
                  ^
SyntaxError: invalid syntax
```

Syntax Error

```
>>> print( 0 / 0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
```

Exception

Exception Handling In Python

Try and Except Block

Try And Except Block

The try and except block in Python is used to catch and handle exceptions

```
try:
    Run this code
except:
    Execute this code when
    there is an exception
```

Try And Except Block

Another example where you open a file and use a built-in exception

```
try:
    with open('file.log') as file:
        read_data = file.read()
except:
    print('Could not open file.log')
```

If file.log does not exist, this block of code will output the following:

OUTPUT

```
Could not open file.log
```

Try And Except Block

In the Python docs, you can see that there are a lot of built-in exceptions that you can use

```
Exception FileNotFoundError

Raised when a file or directory is requested but doesn't exist.
Corresponds to errno ENOENT.
```

```
try:
    with open('file.log') as file:
        read_data = file.read()
except FileNotFoundError as fnf_error:
    print(fnf_error)
```

If file.log does not exist, this block of code will output the following:

OUTPUT

```
[Errno 2] No such file or directory: 'file.log'
```

Different ways to use error handling

```
try - Execution of code (block)
except - block lets you handle the Exception or Error
finally - it always execute
```

```
#way 1

try:
   data="aaa"
   print (data[1])
   print (12/0)
   print ("Hello")
   a=mohammed
except:
   print ("Error Occured")

print (“Technology Academy")
```

```
# way2

try:
   dat=121
   print (dat)
   print ("aa"[1.2])
   print (12/0)

except Exception as e:
   print (e)

print (“Technology Academy")
```

```
#way3

data="Aptech"
try:
   print (data[1.2])
   print (data[6])
   print (12/0)

except IndexError as e:
   print(e)
except TypeError as e:
   print (e)
```

Example

```
'''A module for demonstrating exceptions.'''

def convert(s):
    '''Convert to an integer.'''
    x = int(s)
    return x
```

```
'''A module for demonstrating exceptions.'''

def convert(s):
    '''Convert to an integer.'''
    try:
        x = int(s)
        print("Conversion succeeded! x =", x)
    except ValueError:
        print("Conversion failed!")
        x = -1
    return x
```

```
$ python3
Python 3.3.2 (v3.3.2:d047928ae3f6, May 13 2013, 13:52:24)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more informati
on.
>>> from exceptional import convert
>>> convert("33")
33
>>> convert("hedgehog")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "./exceptional.py", line 5, in convert
    x = int(s)
ValueError: invalid literal for int() with base 10: 'hedgehog'
>>>
```

```
$ python3
Python 3.3.2 (v3.3.2:d047928ae3f6, May 13 2013, 13:52:24)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more informati
on.
>>> from exceptional import convert
>>> convert("34")
Conversion succeeded! x = 34
34
>>> convert("giraffe")
Conversion failed!
-1
>>>
```

Quiz

Quiz 1

Which number is not printed by this code?

- 3
- 2
- 4

```
try:
        print(1)
        print(20/0)
        print(2)
except ZeroDivisionError:
        print(3)
finally:
        print(4)
```

Quiz

2

Fill in the blanks to try to open and read from a file. Print an error message in case of an exception.

```
try:
      [    ] open("test.txt")as [  ]:
       print(f.read())
[    ]
       print("Error")
```

Quiz
3

What is an exception?

- A Variable
- An event that occurs due to incorrect code or input.
- function

Quiz
4

What is the output of this code?

- 1
- 1 3
- 1 2 3
- 3

```
try:
        print(1)
except :
        print(2)
finally:
        print(3)
```