WORKING WITH PYTHON XML PROCESSING

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

Creating XML file or Downloading XML file:

We have created a sample XML file that we are going to parse.

Step 1) Goto google type sample xml file

https://msdn.microsoft.com/en-us/library/ms762271(v=vs.85).aspx

How to Create xml file?

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
</catalog>
```

Save the above XML source code with books.xml @ spyder installed location..!!

Example1:Reading xml file data..!!

```
import xml.etree.cElementTree as ET
tree=ET.ElementTree(file='books.xml')
root=tree.getroot()
for data in root:
  print(data.get('id'))
```

NOTE:

Go to books.xml file, change book as magzine and run the following script.

Example2:Display Book IDs

```
import xml.etree.cElementTree as ET
tree=ET.ElementTree(file='books.xml')
root=tree.getroot()

for data in root:
  if(data.tag=='book'):
    print(data.get('id'))
```

Example3:Display magzine ID..!!

```
import xml.etree.cElementTree as ET
tree=ET.ElementTree(file='books.xml')
root=tree.getroot()

for data in root:
  if(data.tag!='book'):
    print(data.get('id'))
```

Example4: Reading the data from .xml file..!!

```
import xml.etree.cElementTree as ET
tree=ET.ElementTree(file='Books.xml')
root=tree.getroot()

for books in root:
  if(books.tag=='book'):
    print("Book ID is: "+books.get('id'))
    print("----------------------")
    for attr in books:
      if(attr.tag=="author" or attr.tag=="title" or
attr.tag=="price"):
        print(attr.text)
    print("***************")
```

Example5: Writing data in Other file...!!

```
import xml.etree.cElementTree as ET
tree=ET.ElementTree(file='Books.xml')
root=tree.getroot()

for books in root:
  if(books.tag=='book'):
    fo=open(books.get('id') + '.txt','w')
    #print("Book ID is: "+books.get('id'))
    #print("----------------------")
    for attr in books:
      if(attr.tag=="author" or attr.tag=="title" or
attr.tag=="price"):
        #print(attr.text)
    #print("***************")
        fo.write(attr.tag+' : '+attr.text)
        fo.write('\n')
```

WORKING WITH PYTHON JSON

JSON stands for JavaScript Object Notation
JSON is a lightweight data-interchange format
JSON is "self-describing" and easy to understand
JSON is language independent

Why use JSON?

Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.

JSON Syntax Rules

JSON syntax is derived from JavaScript object notation Syntax:

1 Data is in name/value pairs
2 Data is separated by commas
3 Curly braces hold objects
4 Square brackets hold arrays

JSON Uses JavaScript Syntax

Because JSON syntax is derived from JavaScript object notation.!

In JSON, string values must be written with double quotes:

```
{ "name":"Raju" }
```

JSON Files

The file type for JSON files is ".json"
The MIME type for JSON text is "application/json"

JSON vs XML
Both JSON and XML can be used to receive data from a web server.

JSON Example
```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML Example
```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

JSON is Unlike XML Because
1 JSON doesn't use end tag
2 JSON is shorter
3 JSON is quicker to read and write
4 JSON can use arrays

```
$pip install simplejson
$pip show simplejson
$pip install -U simplejson (for Updating)
```

Example:
```
import json
print(dir(json))
```

Example:
```
import simplejson
print(dir(simplejson))
```

Example:
```
book={}
book['raju']={'name':'raju','address':'Ameerpet
HYD','Phone':9866698666}
book['tom']={'name':'tom','address':'5 Green Street
NY','Phone':9866612345}
import json
print(type(book))#<class 'dict'>
s=json.dumps(book)
print(type(s))#<class 'str'>
print(s)
```

Example:

```
book={}
book['raju']={'name':'raju','address':'Ameerpet
HYD','Phone':9866698666}
book['tom']={'name':'tom','address':'5 Green Street
NY','Phone':9866612345}
import json
s=json.dumps(book)
with open("Book.txt",mode='w',encoding='utf-8')as MyFile:
    MyFile.write(s)
```

Example:

```
book={}
book['raju']={'name':'raju','address':'Ameerpet
HYD','Phone':9866698666}
book['tom']={'name':'tom','address':'5 Green Street
NY','Phone':9866612345}
import json
s=json.dumps(book)
with open("Book.txt",mode='r',encoding='utf-8')as MyFile:
    s=MyFile.read()
    print(s)#String
book=json.loads(s)
print(book)#Dictionary
print(type(book))
print(book['raju'])#complete addresss book
print(book['raju']['Phone'])
```

Example:

```
book={}
book['raju']={'name':'raju','address':'Ameerpet
HYD','Phone':9866698666}
book['tom']={'name':'tom','address':'5 Green Street
NY','Phone':9866612345}
import json
print(type(book))
SS=json.dumps(book)#converting dict to string format
print(type(SS))
try:
    with open("Stduent.txt",mode='w',encoding='utf-8') as MyFile:
        MyFile.write(SS)
        print("DataLoadedSuccessfully..!!")
except IOError:
    print("SorryFileUnableToCrete")
finally:
    print("FinallyBlockSuccess..!!")
```

Python Object Serialization

What is Pickle?

Pickle is an inbuilt python library for serializing and de-serializing Python object.

OR

Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening.

OR

Pickling is used to store python objects. This means things like lists, dictionaries, class objects, and more.

Features of Serialization:
1. Python Objects convert into Byte Stream format
2. Over a network byte stream easily transfor
3. Highly secured, good performance
4. Machine easily understandable & Convertable

Storing data with pickle
You can pickle objects with the following data types:
Booleans, Integers, Floats, Complex numbers, Strings, Tuples, Lists, Sets, and Dictionaries etc..!!

Serialization:
It is the process of converting a object hierarchy into a stream of bytes to store in a file.

dump method:
Saves the object to the file stream with the optional parameter of protocol.

Syntax:
```
dump(object,file,protocol)
```

Example: (Pickling)
```
import pickle
data={'a':[1,2,3,4],'b':'NareshIT',(1,2):[1,2]}
with open("Data.pkl",mode="wb") as FPK:
    pickle.dump(data,FPK)
print("PickleFileCratedSuccessfully")
print(data)
del data
print(data)
```

Example:
```
import pickle
PyData={'a':[1,2,3,4],'b':"NareshIT",'c':(1,2)}
try:
    with open("Data.pkl",mode='wb') as FPK:
        pickle.dump(PyData,FPK)
    print("DataSerializedSuccessfully...!!")
except IOError:
    print("SorryDataUnableToSerialize")
finally:
    print("FinallyBlockSuccess")
```

Unpickling:
It is the inverse of Pickling process where a byte stream is converted into an object hierarchy.

load method:
Loads the next object in the file stream and returns the object

Syntax:
```
pickle.load()
```

Example:(De-Serialization)
```
import pickle
```

```
data ={'a':[1,2,3],'b':'NareshIT',(1,2):[1,2]}
with open("Hei.pkl",mode='wb') as FPK:
    pickle.dump(data,FPK)
del data
with open("Hei.pkl",mode='rb')as FPK:
    data=pickle.load(FPK)
print(data)
```

Example:

```
import pickle
try:
    with open("Data.pkl",mode='rb') as FPK:
        PyData=pickle.load(FPK)
    print(PyData)#This is printing in Object format
    print("DataDe-SerializedSuccessfully...!!")
except IOError:
    print("SorryDataUnableToDeserialize")
finally:
    print("FinallyBlockSuccess")
```

Pickle Limitations:

1. It doesn't save objects code, only it's attributes values
2. It cannot store file handles or connection sockets