Big Data Summer Training

BigData Analytics-Python Programming

Prepared and Presenting By: Amrit Chhetri (Certified BigData Analyst),
Principal Techno-Functional Consultant and Principal IT Security Consultant

About Me :

- Me:
  - I'm Amrit Chhetri from Bara Mangwa, West Bengal, India, a beautiful Village/Place in Darjeeling.
  - I am CSCU, CEH, CHFI,CPT, CAD, CPD, IOT & BigData Analyst(University of California), Information Security Specialist(Open University, UK) and Machine Learning Enthusiast (University of California[USA] and Open University[UK]), Certified Cyber Physical System Exert(Open University[UK]) and Certified Smart City Expert.
- Current Position:
  - Principal IT Security Consultant/Instructor, Principal Forensics Investigator and Principal Techno-Functional Consultant/Instructor
  - BigData Consultant to KnowledgeLab
- Experiences:
  - I was J2EE Developer and BI System Architect/Designer of DSS for APL and Disney World
  - I have played the role of BI Evangelist and Pre-Sales Head for BI System* from OST
  - I have worked as Business Intelligence Consultant for national and multi-national companies including HSBC, APL, Disney, Fidedality , LG(India) , Fidelity, BOR(currently ICICI), Reliance Power. * *Top 5 Indian BI System (by NASSCOM)*

Data Science and BigData Processing with Python Training Session-VI

Agendas/Modules:

- Lambda Function
- Python Data Structures
- String Manipulation
- Python External Modules
- Database Programming
- Exception Handling
- File Handling
- Python Regular Expression
- MapReduce using MRJob
- BigData Platform and Python
- Web-Data Programming

Lambda Function:

- Lambda Function is a small and anonymous function and it is defined using lambda
- Lambda function is declared without def keyword
- Lambda can take any number of arguments but returns only one from an expression
- It can access variables on its parameters list and global variables
- Its general syntax: holdervariable=lambda [arguments]:expression
- It is similar to java Ternary operator with same expression for both true and false
- Example:

```
ambdafunc = lambda arg1, arg2, arg3: (arg1*arg2)+arg3

print ("Lambda Computation: ", lambdafunc( 1,2,3 ))

print ("Lambda Computation: ", lambdafunc( 20,30,40))
```

Python Data Structures:

- Numbers, String, List, Tuple and Dictionary are standard Data Types
- List, Tuple, Dictionary and Set are common built-in Data structures in Python.
  - List : Mutable Data Structure and defined using []

    listOS=[“Android”,”Tizen”,”iOS”,”BlackBerry”]
  - Tuple : Immutable Data Structure and defined using ()

    tupleOS=(“Android”,”Tizen”,”iOS”,”BlackBerry”)
  - Dictionary: Key-Value pairs of data, it defined using {}

    dictOS=[1: “Android”,2:”Tizen”,3:”iOS”,4:”BlackBerry”]
  - Sets : An order collection of values, it is defined by {}

    listOS={“Android”,”Tizen”,”iOS”,”BlackBerry”}
- Json returns Python Dictionary or List but not a Tuple

Python Data Structures-List:

- List is a mutable and its common functions are:
  - appen(value) : Appends/adds value at the end
  - min(list) : Returns minimum of the list, works for all data-types
  - max(list) : Returns minimum of the list, works for all data-types
  - sort(reverse=True): Sorts list
  - len(list) : Returns number of elements in the list

- Example:

```
numbers=[]; sum=0
for y in range(0,3):
    localnum=input("Insert Number :") ;  numbers.insert(y,localnum ) ;sum=sum+localnum
numbers.sort(reverse=True)
print(" You entered ( after sorting) :",numbers, "Sum of those numbers",sum) ;numbers.append(100)
print("max:", max(numbers),"min",min(numbers),"Value Count:", len(numbers))
```

Python Data Structures-Tuple:

- Tuple is an immutable Data Structure and its common functions are:
  - list(tuple) : Returns list of elements of a given Tuple
  - index(value) : Returns index of first value
  - count(value): Gives number of occurrence

- Example:

```
aTuple = (2015,2015 ,'Android', 'BlackBerry', 'Ubuntu Mobile');
aList = list(aTuple)
print("A list of Tuple's elements:",aList)
print("Number of values on  Tuple:",aTuple.count(2015))
print("The position of 2015",aTuple.index(2015, ))
```

- Empty Tuple is declared as tupleA=()

Python Data Structures-Dictionary:

- Dictionary is key-value Data Structure and its common functions are:
  - len(dictionary): Gives number of key-value mapping
  - keys() : Returns keys as a list
  - get(key) : Retrieves for a given key
- Example:

```
mapValue={30:"No Strings Attached", 100:"The Bone Collector",60:"Love From France"}
print("Dictionary",mapValue) ;keys=[] ;keys=mapValue.keys()
keys.sort(reverse=True) ;print("All Keys", keys, len(mapValue))
for key in keys:
        print(key,mapValue.get(key))
```

- Empty Dictionary is declared as dictBlank={} and value is assigned as dictBlank['key1']="Android"

String Manipulation:

- sys.stdin.readline() function can take strings with spaces as user inputs
- Comments are used to improve the readability of codes and they are:
  - Single Line : # Single Line comment
  - Multiple Line: “”” Multi-lines comments “””
- Example of reading String inputs:

```
import sys
def takeAndPrint():
    name=[]
for x in range(0,1):
   print("Enter name Line - "+str(x+1)+":") ;  name.append((sys.stdin.readline()).capitalize()) ;print(name)
takeAndPrint()
```

String Manipulation-Functions:

- Python has good number functions for String manipulations, they are :
  - plit(spliter) : Returns a list of words spliting by given character
  - len(string) : Returns number of characters including special characters
  - index(string) : Gives the index/position of first occurrence
  - startswith(string): Indicates whether as string starts with given substring
  - isdigit() : Returns true is all characters are digits
- Example of String Functions:

```
text="Dr. No is one of my favorite movies“ ; print("Extracting as List element:",text[0:len(text)])
if(text.startswith("Dr.")):
   print("Movie text is at :",text.index("movies"))
def findEmails():
    text= "Me, Amrit Chhetri is IoT Security geek reachable at  amritchhetrib@ymail.com“ ;
    print("Email:",text[len(text)-23: len(text)] )
findEmails()
```

Python External Modules:

- Python external modules can be install using different options and pip is one of them.
- Steps for pip:
  - Install pip using get-pip.py , python get-pip.py and creates/updates scripts folders ;
  - Run pip install <module name>
- Common External Modules:
  - Sqlite3 for SQLite Database: pip install pysqlite or pip install sqlite
  - Mysqldb for MySQL Database : pip install MySQL-python or pip install pymysql
  - Mysqldb (Linux) : apt-get install python-pip ; apt-get install python-dev libmysqlclient-dev
  - MS SQL Server : pip install pymssql
  - BeatifulSoup : pip install beautifulsoup4
  - Py2exe : pip install setuptools , pip install y2exe ; Json module: pip install simplejson

Database Programming:

- Python support Database Connectivity to all standards Databases including MongoDB.
- External Python Modules are installed to Programs for Database
- Python Databases Modules:
  - SQL: mysqldb
  - SQLite : sqlite3
  - MongoDB: pymongo
  - MS SQL Server: mssqldb
- Thrift Module is used to connect to Hive Server in BigData Ecosystem

Exception Handling:

- Exception Handling and Assertions are two features for Error Handling and Debugging. Assertions is a sanity-check that can be turn on or turn-off
- try-except-finally is used for exception handling
- Example:

```
try:
        fi = open("E:\\writeSum.txt", "r")
        line = fi. readline() ; numbers=[]; prd=1;numbers=line.split(",")
        for x in numbers:
            prd=prd*int(x)
        print("Product: %s" % str(prd))
except:
        print("IO Exception")
finally:
        fi. close()
```

Python Regular Expressions:

- Regular expression is UNIX-style expression using character sequence
- Regular expression is achieved by importing re module
- The common function are:
  - re.match(pattern, string, flag=0) ; re.search(pattern, string)
  - re.findall(pattern, string)
- Example:

```
import re
fo = open("data.txt", "r") ;    line = fo. readline();    words=line.split(" ")
for word in words:
   if re.search("Data", word):
      print("Data is there")
   else:
      print("Data Not Found")
```

MapReduce using MRJob:

- mrjob allows to write MapReduce in Python with three functions-mapper(), reducer() and combiner()
- Example:

```
from mrjob.job import MRJob ; import re

expression= re.compile(r"[\w']+")

class WordCount(MRJob):

    def mapper(self, _, line):

        for word in expression.findall(line):

            yield (word.lower(), 1)

    def combiner(self, word, counts):

        yield (word, sum(counts))

    def reducer(self, word, counts):

        yield (word, sum(counts))

if __name__ == '__main__':

    WordCount.run()
```

BigData Platform and Python:

- Hadoop and Python goes well in BigData ecosystem
- Inside BigData , Python are used for
  - MapReduce : mrjob, pydoop
  - Analytics : pySpark
  - Web Data Streaming : tweepy, BeautifulSoup, urllib,json
  - GUI Building : PyQt4, Tkinter, WxPython
  - Database Programming: python-mysql, sqlite3
- 'Python for Data Science' is a great book for BigData Programming with Python

Web Data Programming:

- Python supports Web Data Programming and common modules for it are:
  - json, BeautifulSoup, urllib,twepy,pydoop
- Python json libray parses JSON from string or file
- Json produces python dictionary,{}, or list,[]
- It also does reverse, converts list or dictionary to json
- JSON Example:

```
import json
 json_data = '{"name":"Amrit Chhetri","Hobbies":" IT Security,PenTest,BigData,IoT,CPS,Cyber Forensics
 Psychology"}'
 parsed_json = json.loads(json_data)
 print(parsed_json['name']+"'s Hobbies :" + parsed_json['Hobbies'] )
```

THANK YOU ALL