JYTHON

Jython is the JVM implementation of the Python programming language. It is designed to run on the Java platform. Jython was created in 1997 by Jim Hugunin.

A Jython program can import and use any Java class. Just as Java, Jython program compiles to bytecode. One of the main advantages is that a user interface designed in Python can use GUI elements of AWT, Swing or SWT Package.

Difference between Python and Jython:

1. Standard Python is available on multiple platforms. Jython is available for any platform with a JVM installed on it.
2. Standard Python code compiles to a **.pyc** file, while Jython program compiles to a **.class** file.
3. Python extensions can be written in C language. Extensions for Jython are written in Java.

Installation:

Jython is available in the form of an executable jar file.

run the following command –

```
java -jar jython_installer-2.7.0.jar
```

An installation wizard will commence with which installation options have to be given. Then follow the given steps

After the installation is complete, invoke **jython.exe** from the bin directory inside the destination directory. Assuming that Jython is installed in **C:\jython27**, execute the following from the command line.

```
\jython27\bin\jython
```

A Python prompt (>>>) will appear, in front of which any Python statement or Python script can be executed.

One of the most important features of Jython is its ability to import Java classes in a Python program. We can import any java package or class in Jython, just as we do in a Java program. The following example shows how the **java.util** packages are imported in Python (Jython) script to declare an object of the Date class.

```
from java.util import Date

d = date()

print d
```

Save and run the above code as **UtilDate.py** from the command line. Instance of the current date and time will be displayed.

```
C:\jython27\bin>jython UtilDate.py
Sun Jul 12 00:05:43 IST 2020
```

Any Java package for that matter can be imported in a Jython script. Here, the following java program is stored and compiled in a package called **foo**.

```
Package foo;
  Public class HelloWorld{
    Public void hello() {
      System.out.println(“Hello World”);
    }
  Public void hello(String name){
    System.out.printf(“Hello %s!”, name);
  }
}
```

This **HelloWorld.class** is imported in the following Jython Script. Methods in this class can be called from the Jython script **importex.py**.

```
from foo import HelloWorld

h = HelloWorld()

h.hello()

h.hello(“Hello world”)
```

Save and execute the above script from the command line to get following output.

```
C:\jython27\bin>jython importex.py
Hello World!
Hello hello world
```

Basic syntax:

Variables are named locations in computer's memory. Each variable can hold one piece of data in it. Unlike Java, Python is a dynamically typed language. Hence while using Jython also; prior declaration of data type of variable is not done. Rather than the type of variable deciding which data can be stored in it, the data decides the type of variable.

```
> x=10

>> type(x)

<class  'int'
```

- List
- Tuple
- Dictionary

In addition to Python's built-in data types, Jython has the benefit of using Java collection classes by importing the **java.util package**. The following code describes the classes given below –

```
import java.util.ArrayList as ArrayList
arr = ArrayList()
arr.add(10)
arr.add(20)
print "ArrayList:",arr
arr.remove(10) #remove 10 from arraylist
arr.add(0,5) #add 5 at 0th index
print "ArrayList:",arr
print "element at index 1:",arr.get(1) #retrieve item at index 1
arr.set(0,100) #set item at 0th index to 100
print "ArrayList:",arr
```

The above Jython script produces the following output –

```
C:\jython27\bin>jython arrlist.py

ArrayList: [10, 20]

ArrayList: [5, 20]

 element at index 1: 20

 ArrayList: [100, 20]
```

The following code block is a Java program having an embedded Jython script **"hello.py".usingexecfile()** method of the PythonInterpreter object. It also shows how a Python variable can be set or read using set() and get() methods.

```
import org.python.util.PythonInterpreter;
 import org.python.core.*;

 public class SimpleEmbedded {
     public static void main(String []args) throws PyException {
     PythonInterpreter interp = new PythonInterpreter();
      System.out.println("Hello, world from Java");
      interp.execfile("hello.py");
     interp.set("a", new PyInteger(42));
     interp.exec("print a");
     interp.exec("x = 2+2");
      PyObject x = interp.get("x");
     System.out.println("x: "+x);
     System.out.println("Goodbye ");
      }
 }
```

```
Hello, world from Java
hello world from Python
 42
 x: 4
 Goodbye
```

Advantages:

1. Run Python on Java Platform
2. Jython enables programmers to run Python code on any Java Virtual Machine (JVM). They can compile the source code written in Python to Java bytecode, and run the bytecode on any JVM. The implementation allows them to integrate a dynamic and high-level programming language like Python with the hugely popular Java platform.
3. Reduces Coding Time Significantly
4. The simple syntax rules of Python enable programmers to express concepts and accomplish programming tasks without writing longer lines of code.
5. Take Advantage of Most Java Classes
6. Jython also borrows the object-oriented programming features of Python. Thus, it becomes easier for programmers to import and use a wide variety of Java classes inside Python code.
7. Provides an Interactive Interpreter
8. Jython further provides programmers with an interactive interpreter to make their application interact with Java packages and applications.

Disadvantages:

1. It is really difficult to suggest jython for analytics and statistics operations e.g. pandas, sklearn, numpy, scipy depended taks.

2. Restrictions of compulsory JVM availability on remote machines.