**ASSIGNMENT 4**

1. **What is Cpython and Jpython?**

**CPython**

The default and most widely used compiler of python. **Cython** or CPython is an interpreter has its pillars designed in C Language which had its stable release of Python 3.6.5 in March 2018 and for Python 2.7 in May 2018.

Cython is an **optimising static compiler** for both the **Python** programming language and the extended Cython programming language (based on **Pyrex**). It makes writing C extensions for Python as easy as Python itself.

CPython has an ability to provide combined features of C and Python like:

* Python code can easily be written that calls back and forth from and to C or C++ code natively at any point
* easily tune readable Python code into plain C performance by adding static type declarations, also in Python syntax.
* use combined source code level debugging to find bugs in your Python, Cython and C code.
* interact efficiently with large data sets, e.g. using multi-dimensional NumPy arrays.
* quickly build your applications within the large, mature and widely used CPython ecosystem.
* integrate natively with existing code and data from legacy, low-level or high-performance libraries and applications.

The Cython language is a superset of the **Python** language that additionally supports calling **C functions** and declaring **C types** on variables and class attributes. This allows the compiler to generate very **efficient C code** from Cython code. The C code is **generated once** and then compiles with all major C/C++ compilers in CPython 2.6, 2.7 (2.4+ with Cython 0.20.x) as well as 3.3 and all later versions

**Limitations of CPython:**

* Each CPython interpreter for Python, the process uses a GIL(Global Interpreter Lock). This serves as a limitation as it disables concurrent Python threads for a process.
* Another problem is that to achieve concurrency, you must manage separate CPython interpreter processes with a multitasking OS. This also makes it harder for concurrent CPython processes to communicate.

**JPython**

JPython or Jython is successor of CPython and has its pillars build on the Java Platform. It had its stable release for Python 2.7.1 in July 2017 and had its first release in January 2001.

Jython takes Python code and compiles it to Java bytecode. This means we can run Python on any machine that runs a **JVM (Java Virtual Machine)**. Jython supports static and dynamic compilation and let’s extend**Java classes**.

Jython is a Java implementation of Python that combines expressive power with clarity. Jython is freely available for both commercial and non-commercial use and is distributed with source code under the PSF License v2. Jython is complementary to Java and is especially suited for the following tasks:

* **Embedded scripting -** Java programmers can add the Jython libraries to their system to allow end users to write simple or complicated scripts that add functionality to the application.
* **Interactive experimentation** - Jython provides an interactive interpreter that can be used to interact with Java packages or with running Java applications. This allows programmers to experiment and debug any Java system using Jython.
* **Rapid application development** - Python programs are typically 2-10x shorter than the equivalent Java program. This translates directly to increased programmer productivity. The seamless interaction between Python and Java allows developers to freely mix the two languages both during development and in shipping products.

1. **Difference between Python 2 and Python 3 ?**

|  |  |
| --- | --- |
| **Python 2** | **Python 3** |
| * 1. Python 2 had its first release in October, 2000 and it was followed by its successor versions till Python 2.7 which was released in July 2010. | Python 3 had its first release in December, 2008 followed by its successor versions with the latest stable release of python 3.7. |
| * 1. It is more stable and transparent version of python programming language. | It is the future of Python designed to address the design flaws in the previous versions. |
| * 1. The print-syntax is treated as a statement rather than a function which requires text to be wrapped in parenthesis. | The print is explicitly treated as a function and replaces by the print() function in python3 which requires an extra pair of parenthesis. |
| * 1. ASCII string type is used by default to store strings. | Unicode is the implicit string type by default |
| * 1. It simply returns an integer to the nearest whole number when dividing two integers. | It makes integer division more intuitive using true division for integers and floats. |
| * 1. xrange() function reconstructs the sequence every time. | Xrange is replaced by rnage() function in python 3. |
| * 1. In python2 there were two methods to take input: * Raw\_input() * Input() | In python3 raw\_input() was removed and there is only one input() method left. |
| * 1. It had next() function and .next() method for iterate next element. | Have only next() function to iterate. |

1. **Difference between ASCII and Unicode?**

* ASCII defines 128 characters, which map to the numbers 0–127. Unicode defines (less than) 221 characters, which, similarly, map to numbers 0–221 (though not all numbers are currently assigned, and some are reserved).
* Unicode is a superset of ASCII, and the numbers 0–127 have the same meaning in ASCII as they have in Unicode. For example, the number 65 means "Latin capital 'A'".
* Because Unicode characters don't generally fit into one 8-bit byte, there are numerous ways of storing Unicode characters in byte sequences, such as UTF-32 and UTF-8.