Introduction to Python

Programming languages like C, Pascal or Fortran concentrate more on the functional aspects of programming. In these languages, there will be more focus on writing the code using functions. for eg. we can imagine a C program as a combination of several functions. computer scientists thought that programming will become easy for human beings to understand if it is based on real life examples. Hence they developed Object Oriented Programming Languages like Java and .NET where programming is done through classes and objects. Programmers started migrating from C to Java and Java soon became the most popular language in software community.

In Java, a programmer should express his logic through classes and objects only. It is not possible to write a program without writing at least one class!

This makes programming lengthy.
For example, a simple program to add two numbers in JAVA looks like this.

```
// Java program to add two numbers
class Add        // create a class
{  public static void main(String args[])
   {
      int a, b ;
      a = b = 10
      system.out.println ("sum =  "+ (a+b));
   }
}
```

Programmers understood that in certain cases where there is no need to go for classes or objects, this type of coding is consuming more time. In such cases, they do not want to create classes or objects; rather they want to write C style coding. The same program to add two numbers can be written in C as:

```
# include<stdio.h>
void main()
{   int a, b;
    a = b = 10;
    printf("sum = %d", (a+b));
}
```

The preceding program is almost same as that of Java. There is no improvement in the length of code. Another problem is that if the programmers go for C language, they will miss the object orientation which is lacking in C.

Why Python Required ?

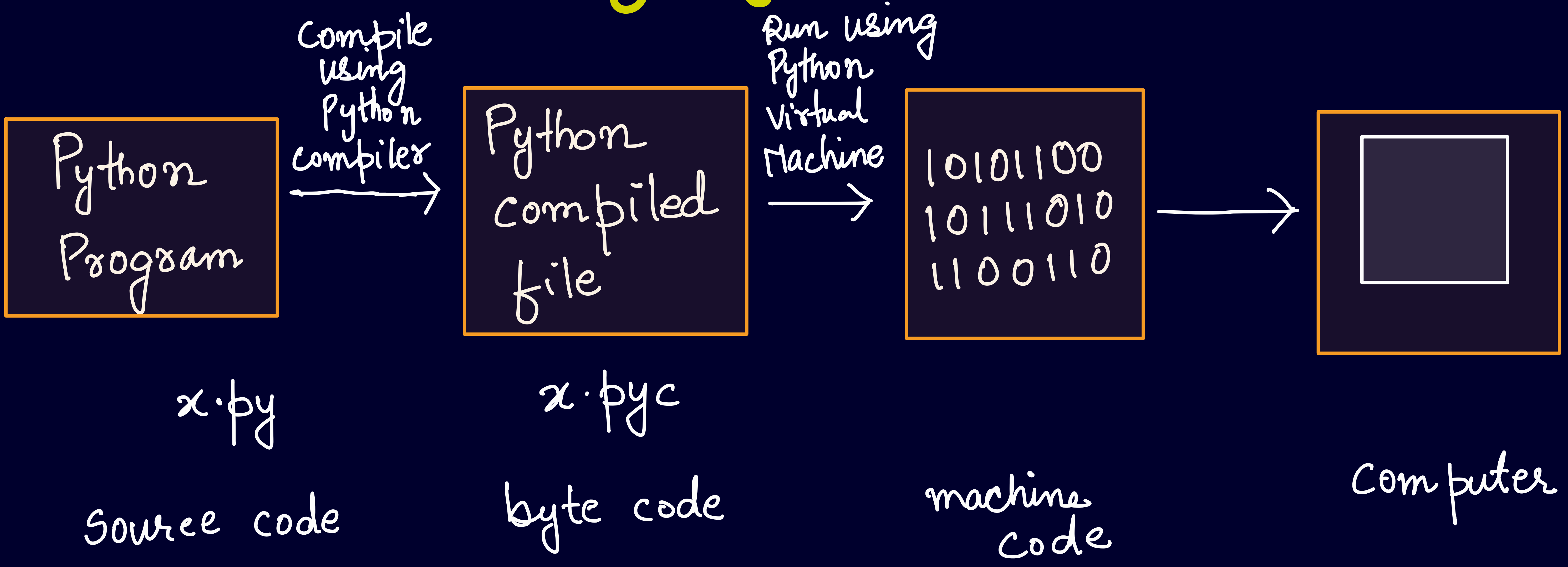
→ Programmers want C style coding (simple style) as well as the Java style object orientation. like when they want to develop functional aspects like calculations or processing, they want to use C style coding and when they are in need of going for classes and objects, they will use Java style coding. The only answer for their requirement is PYTHON !

Features of Python

- Simple
- Easy to Learn
- Open source
- High level language
- Dynamically Typed
- Platform independent
- Portable
- Procedural
- Object Oriented
- Interpreted
- Extensible
- Embeddable

- Huge Library
- Scripting language
- Database connectivity
- Scalable
- Batteries included

Execution of Python



Viewing the Byte Code

Lets considered the following Python Program

```
a = b = 10
print ("sum : ", (a+b))
```

We can type this program in a text editor like Notepad and then save it as first.py. It means the first.py file contains the source code.

Now lets compile the program using python compiler

```
C:\> python first.py
```

it will display the result as :

```
Sum = 20
```

That is OK. But we do not want the output of the program. We want to see the byte code instructions that were created internally by Python compiler before they are executed by PVM.

for this purpose, we should specify the dis module while using python command as :

```
c:\>python -m dis first.py
```

It will produce the following output:

```
2     0   LOAD_CONST       0 (10)
      3   DUP_TOP
      4   STORE_NAME       0 (a)
      7   STORE_NAME       1 (b)

3    10   LOAD_NAME        2 (print)
     13   LOAD_CONST       1 ('sum =   ')
     16   LOAD_NAME        0 (a)
     19   LOAD_NAME        1 (b)
     22   BINARY_ADD
     23   CALL_FUNCTION    2 (2 positional)
```

The preceding byte code is displayed by the dis module, which is also known as disassembler. that displays the byte code in human understandable format. If we observe the preceding code, we can find 5 columns.

The left most or first column represents the line number in our source program. The second column represents the offset position of byte code. The third column shows the name of the byte code instruction. The 4th column represents instructions argument and the last column represents constants or name as specified by 4th column. For example, see the following instructions:

```
10    LOAD_NAME     2 (print)
13    LOAD_CONST    1 ('sum = ')
```

The LOAD_NAME specifies that this byte code instruction has 2 arguments. The name that has 2 arguments is (print) function.
Since there are 2 arguments, the next byte code instructions will represents those 2 arguments as

LOAD_CONST represents the string constant name ('sum = ') and (a) and (b) as names involved in the second argument for the print function.
Then BINARY_ADD instruction adds the previous (i.e a and b) values.

Flavors of Python

Flavors of Python refer to the different types of Python compilers. These flavors are useful to integrate various programming languages into Python. The following are some of them:

- C Python
- Jython
- Iron Python
- Py Py
- Ruby Python
- Stackless Python
- Pythonxy
- Anaconda Python