How to Create Programs?

A basic introduction to earlier generation programming languages and a small comparison of syntax complexity between the latest technologies being used.

Closest to the Hardware: Assembly Language

The only programming language directly understandable by a computer is machine language. A more human-readable representation of machine language is assembly language. It is a set of very primitive operations linked to a specific family of processors (the computer's "brain") and manipulating its memory.

Here's an example of a basic program written in assembly language. It displays `"Hello"` to the user.

```
str:
.ascii "Hello\n"
.global _start

_start:
movl $4, %eax
movl $1, %ebx
movl $str, %ecx
movl $8, %edx
int $0x80
movl $1, %eax
```

```
movl $0, %ebx
int $0x80
```

Pretty scary, isn't it? Fortunately, other programming languages are much simpler and convenient to use than assembly language.

The Family of Programming Languages

There are a large number of programming languages, each adapted to different uses and with its own syntax. However, there are similarities between the most popular programming languages. For example, here's a simple program written in Python:

```python
print("Hello")
```

You can also write the same thing in PHP:

```php
<?php
echo("Hello\n");
?>
```

Or even C#!

```csharp
class Program {
static void Main(string[] args) {
        Console.WriteLine("Hello");
    }
}
```

What about Java?

```
public class Program {
    public static void main(String[] args) {
        System.out.println("Hello");
    }
}
```

All these programs display `"Hello"` through a different set of instructions.

Program Execution

The fact of asking a computer to process the orders contained in a program is called *execution*. Regardless of which programming language is used, a program must be translated into assembly code in order to be executed. The translation process depends on the language used.

With some languages, the translation into assembly code happens line by line in real time. In this case, the program is executed like a human reads a book, starting at the top and working down line-by-line. These languages are said to be *interpreted*. Python and PHP are examples of interpreted languages.

Another possibility is to read and check for errors throughout the whole source code before execution. If no errors are detected, an executable targeting one specific hardware platform is generated. The intermediate step is called compilation, and the programming languages which use it are said to be *compiled*.

Lastly, some languages are pseudo-compiled in order to be executed on different hardware platforms. This is the case for the Java language and also for those of the Microsoft .NET family (VB.NET, C#, etc).