Theory: Program execution

9 minutes   9 / 12 problems solved

Start practicing

7633 users solved this topic. Latest completion was about 1 hour ago.

Current topic:

✓ Program execution Stage 1 3

Topic depends on:

✓ Multi-line programs Stage 1 18

Topic is required for:

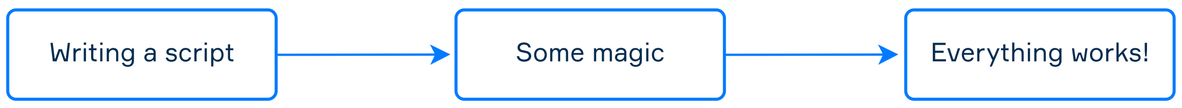
✓ Errors Stage 1 3

Have you ever thought about what it means to write a program in Python? From the **programmer's point of view**, it simply means to write a set of familiar Python statements in a text file and later make Python execute the file. Thus, one can create a .txt file with this statement:

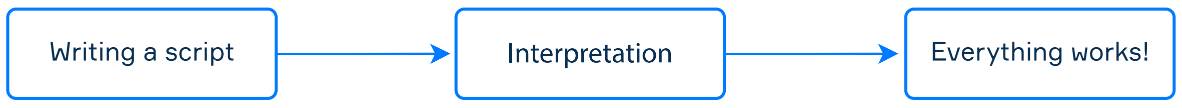
```
print("Hello, World!")
```

and run it via Python or OS console. Of course, by convention, all the Python files are supposed to be called with **.py** extension and typically people don't use text editors to write Python code — they use **IDE**s, but the idea is clear: from the programmer's point of view source code is just a **set of statements**. But that's not exactly it.

If you've been programming for at least some time, you probably have heard such terms as **interpreted** or **compiled** languages. And most likely you've heard that **Python is an interpreted one**, without any details. So, let's figure this all out!
Right now, the process of program execution must look for you something like this:



In fact, taking into account the part with "**interpretation**" process, you can turn on your logical thinking and change the middle part:



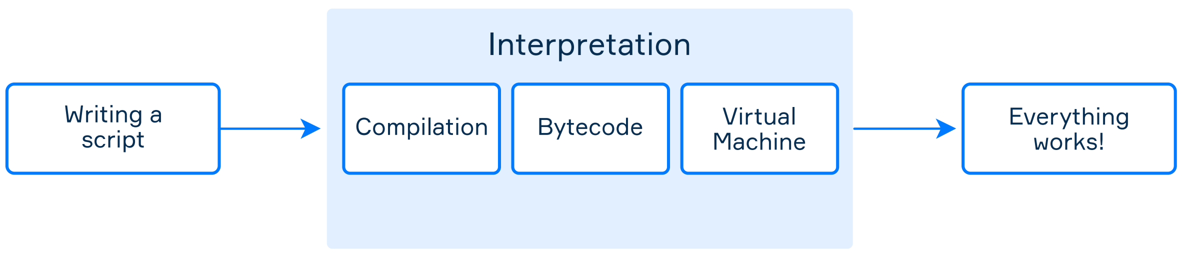
...and that'll do. Actually, the majority of Python programmers stop going deeper at this step. It's usually enough to know that "**Python is an interpreted language**" and to explain anything using this mantra. Here we are going to be better — we'll go deeper!

§1. The interpreter

Generally, what is the interpretation process? One may say that it resembles the **reading** of a program. Some software just "reads" your program and executes what is written in it **line by line**. This software is unexpectedly called **Interpreter** and is a part of the standard Python installation package. Another inherent part of it is the **standard library** with all built-in modules, functions, data, etc. An interesting fact: an interpreter can be written in basically any programming language. The default interpreter for Python is written in C and is called **CPython**. Some other Python interpreters are:

- **PyPy** is written in a restricted subset of Python called RPython (Restricted Python), which provides some restrictions to the usual Python code.
- **Jython** translates Python code into Java-compatible byte code, which is later executed by JVM — Java Virtual Machine.
- **IronPython** is an implementation of Python for the .NET framework.

Now let's try to understand, what is really happening during interpretation. In fact, this step consists of 3 smaller ones:



The compiler turns your set of statements (your source code) into so-called **byte code**. Byte code itself is **lower level** (thus more detailed), **platform-independent**, and **more efficient** version of source code, but it's **not binary machine code** like instructions for an Intel or AMD chip. Byte code is a **Python-specific** representation of source code. That's why some Python programs are executed not as fast as the analogs in C++ or C — traditional compiled languages.

§2. Python Virtual Machine

After compilation, the byte code is given into the **PVM (Python Virtual Machine)**. Although it sounds quite impressive, in fact, it is nothing more than a big piece of code, that iterates through byte code instructions received from a compiler and executes them **one by one**, thus performing the desirable operations specified by the programmer. It is an internal part of the Python system and you don't need to install it as it's not a separate program. In fact, the thing which really executes your code is PVM, so we can say it is the last step of executing any Python program. All this complexity is deliberately hidden from a programmer. The "interpretation part" is **fully automated**, so usually you won't need to think about it. Remember: Python programmers simply write the code and run the files, everything else is done by Python itself.

§3. Conclusion

Now you should understand that yes, Python is an interpreted language indeed, but before the interpretation stage, there is an internal process of compiling the source code into the byte code. Thus, executing a Python program implies both steps: compilation and interpretation.

When your program is turned into byte-code, it's executed **line by line from top to bottom**, so generally you can expect your program to work exactly the way you have written it.

Report a typo

682 users liked this theory. 14 didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

This content was created over 1 year ago and updated about 12 hours ago. Share your feedback below in comments to help us improve it!

Comments (22) Hints (0) Useful links (1) Show discussion