10118 users solved this topic. Latest completion was 7 minutes ago.

Theory: Program execution

10 minutes 0 / 0 problems solved

Skip this topic Start practicing

What does it mean to write a program in Python? From the programmer's point of view, it implies writing certain instructions in a file and then executing it in Python. We can simply create a *.txt* file with the following statement:

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

And then execute it from the command line in the following way (suppose, we named it *%example.txt%*):

```
python example.txt
```

You may know that all Python files are supposed to have a *.py* extension, but we can write the code in a *.txt* file as well. Usually, programmers don't use text editors for programming, they use **IDE**s (Integrated Development Environment), a special program designed for software development. However, one thing remains — the source code is just a set of specific instructions.

It's not all that simple. You may have heard that there are **interpreted** and **compiled** programming languages. Maybe you've also heard that Python is an interpreted language. But do you know what that means?

§1. Interpretation

The process of program execution may seem something like this to you:

Writing a script
Some magic
Everything works!

The **interpretation** part, in fact, fits into the middle part of this scheme:

Writing a script
Interpretation
Everything works!

The majority of Python programmers just stop inquiring further from this point. Let's try to explain what the interpretation means in this case.

During the interpretation part, the software goes through your program and executes the code, line by line. This software is called an **interpreter**. It is a part of the standard Python installation package. So, an interpreter is a layer of software logic between your source code and your hardware.

An interpreter can be written in any programming language. The default Python interpreter is written in C, it is called CPython. This is what we install from the official Python website.

§2. Interpreted or compiled?

To understand what is under the hood of Python, let's find out what lies behind the concept of interpreted and compiled languages.

With compiled languages, when a programmer wants to run a source code, they need to use a **compiler** first. It is the special software that translates the source code into a special language. This language is called the **machine code**. It is a set of instructions executed directly by your processor. The problem is that different types of computers operate with different machine codes, so the source code compiled in Windows may not work on a Mac.

Current topic:

Program execution Stage 1

Topic depends on:

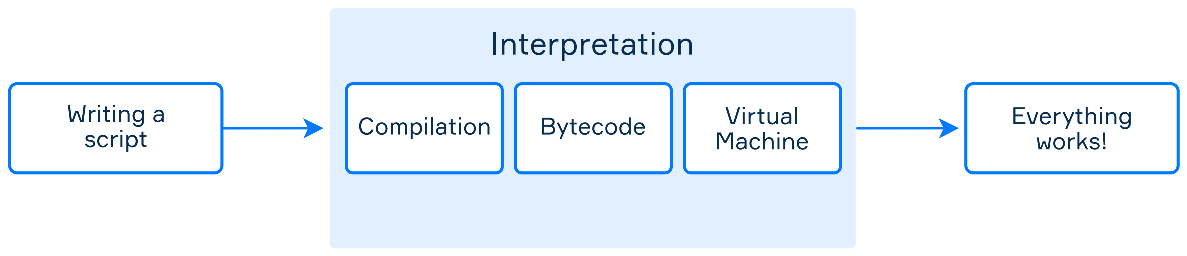
✓ Multi-line programs 11★ Stage 1

Topic is required for:

Errors Stage 2

Interpreted languages also include a compilation step, but the compiler turns your source code into the so-called **byte code**, an intermediate language. The byte code is a lower level (more detailed), platform-independent, and more efficient version of the source code. We need a special abstract computer (the Python Virtual Machine in our case, but more on that later!) to execute this code.

In fact, the interpretation step consists of three phases:



Byte code is saved as a *.pyc* file after the compilation. If the source code of the program has not been changed since the last compilation, Python will load the existing *.pyc* file, bypassing the compilation step.

Below you can see the readable byte code. The second column contains the instructions in the order of their execution. If something is unclear, it's OK. You don't have to know the byte code inside out to write programs.

```
1           0 LOAD_NAME                0 (print)
            2 LOAD_CONST               0 ('Hello, World!')
            4 CALL_FUNCTION            1
            6 RETURN_VALUE
```

§3. Python Virtual Machine

The byte code is supplied to the PVM (Python Virtual Machine) after the compilation. It may sound quite impressive, but in fact, it is nothing more than a big piece of code that iterates through the byte code instructions, which were received from a compiler, and executes them one by one. It is an internal part of the Python system and you don't need to install it as a separate program.

PVM in CPython does not convert the byte code into the machine code. It can execute it right away.

This complexity is hidden from a programmer for a reason. The interpretation is fully automated, and you won't need to think about it at all. Remember that Python programmers simply write the code and run the files, Python does everything else.

§4. Conclusion

Python is an interpreted language. It compiles the source code to the byte code before the execution. Executing a Python program implies both compilation and interpretation.

When your program is turned into the byte code, it's executed line by line by the PVM. That's why Python is slower than compiled languages such as C++ or C.

Report a typo

883 users liked this piece of theory.. 18 didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Comments (25)   Hints (0)   Useful links (2)   Show discussion