Language execution model

Python interpreter

- Python is implemented mostly as an **interpreted** language, but part of the execution process is actually **code compilation**.
- Being interpreted means that the Python code you write is input to an **interpreter** program that executes the code as it reads it **line after line** (usually python source code is saved in *.py* files)

    - This interpretation process is distictively different from compilation (in languages like C or Java) where the language compiler translates initially the code into compiled form (a form of code closer to what the machine 'understands').

- Pros/cons of interpreted implementation:

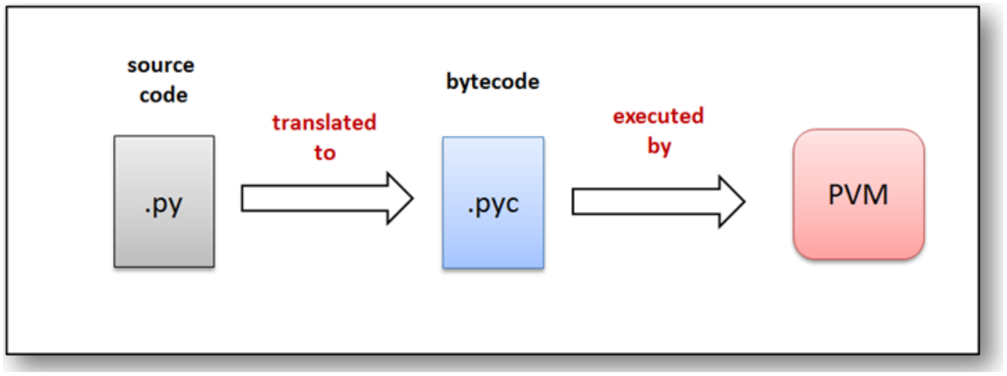
    - *Pros*: interpretation offers a **more interactive learning experience** to the beginner, as it immediately executes the code and provides feedback
    - *Cons*: interpretation **tends to be slower in code execution** since an interpreter reads (parses), interprets and executes source code 'from scratch' each time (no compiled form of code is created).

Bytecode and Python Virtual Machine (PVM)

- Python, however, is not a simply interpreted language. Code execution is somehow more complicated.
- In essence, Python creates an intermediate translated form of code called **'bytecode'**. This is platform independent code (usually in *.pyc* files) which is executed by a special form of Python program called **"Python Virtual Machine" (PVM)**
- Since executing bytecode is faster than executing source code from scratch Python tends to be faster than purely interpreted implementation.
- When working with Python the problem of interpreted-code slower execution is alleviated by at least two factors:

> - (a) Contemporary hardware is fast enough to efficiently execute Python bytecode in many situations
> - (b) In time critical situations there is always the possibility of "mixing" Python with faster executed compiled C code.

Actually the basic Python you learn is itself written in C ('CPython' implementation), meaning that many of its modules call C compiled code routines, thus achieving C comparable execution times.



Python language core execution model
