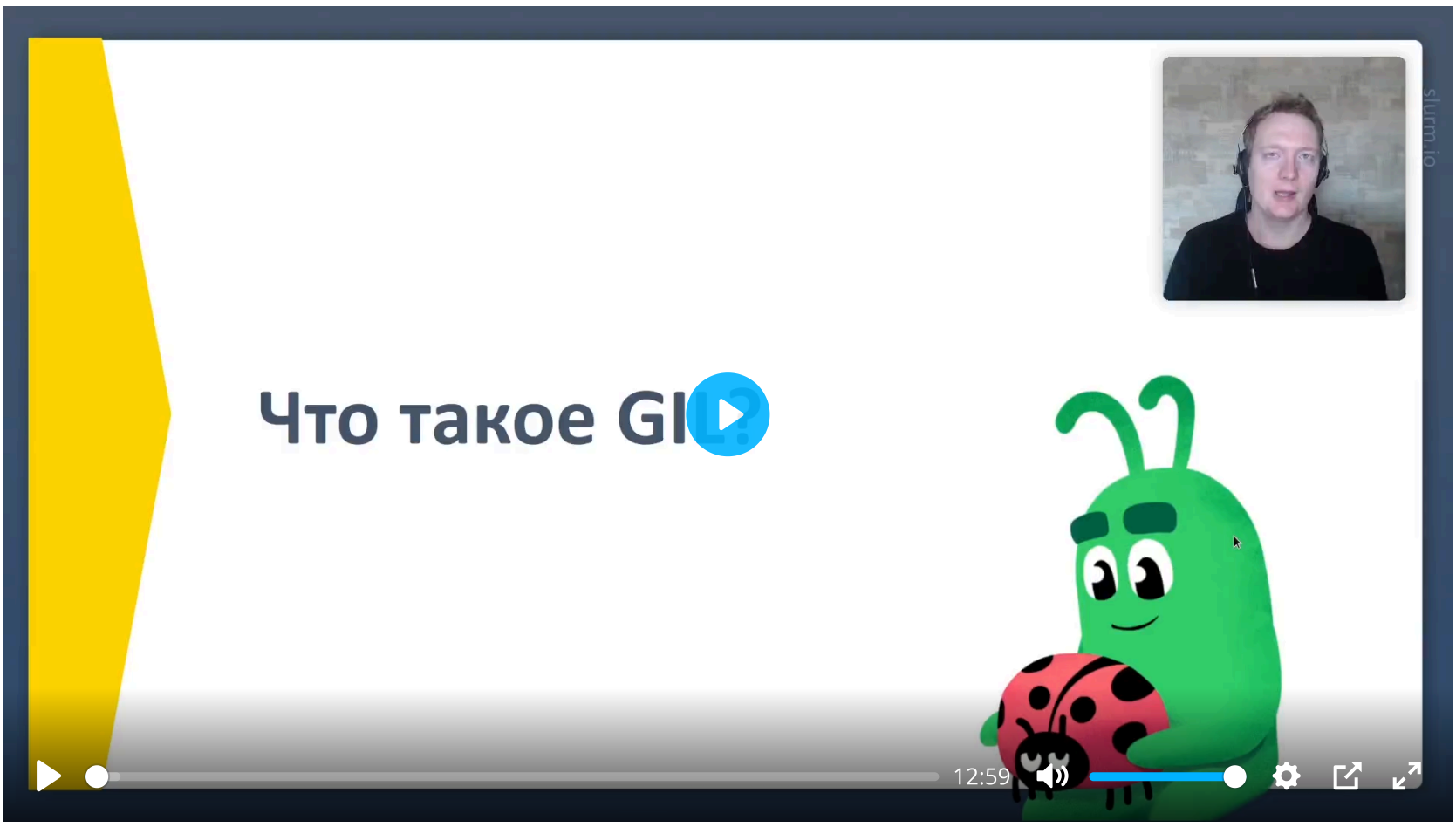
2.3 Параллельность и асинхронность в Python



Текстовая расшифровка видео:

ЧТО ТАКОЕ GIL?

План:

- GIL;
- Ассемблер или байткод;

GIL

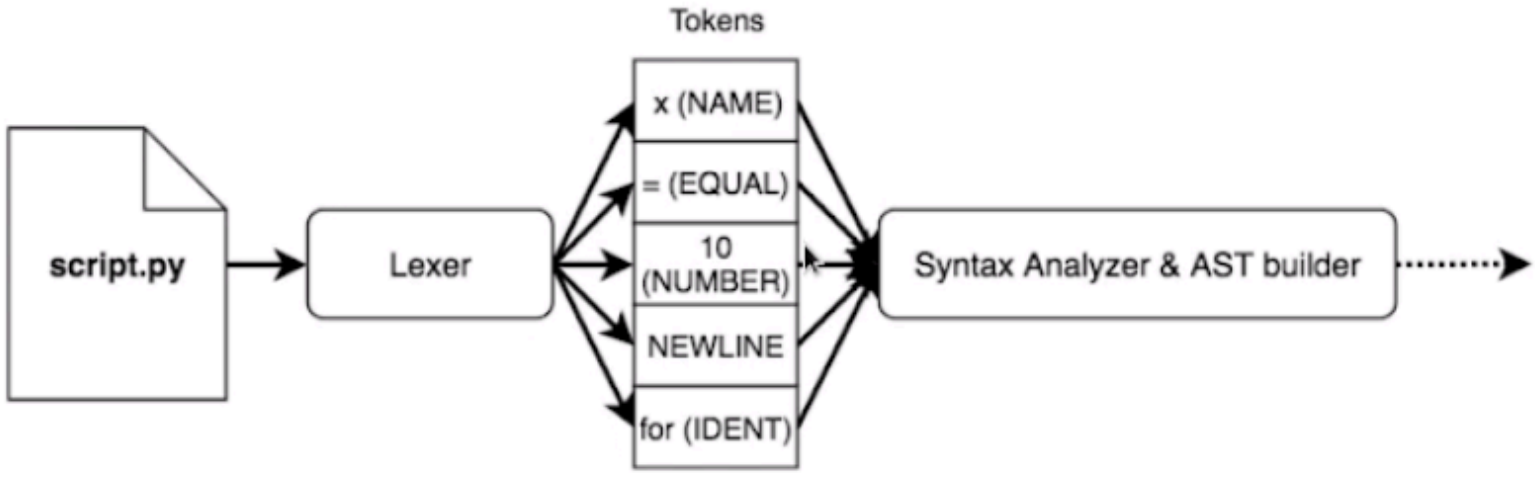
GIL (Global Interpreter Lock) – это глобальный мьютекс (механизм синхронизации в интерпретаторе Python.

Мьютекс – это примитив синхронизации, который обеспечивает взаимное исключение исполнения критических участков кода.

GIL запрещает выполнять байткод Python **больше, чем одному потоку одновременно**. Это касается только байткода Python и не распространяется на I/O операции.

Потоки Python (в отличие от потоков в некоторых версиях Ruby) – это полноценные потоки ОС.

Рассмотрим пример:

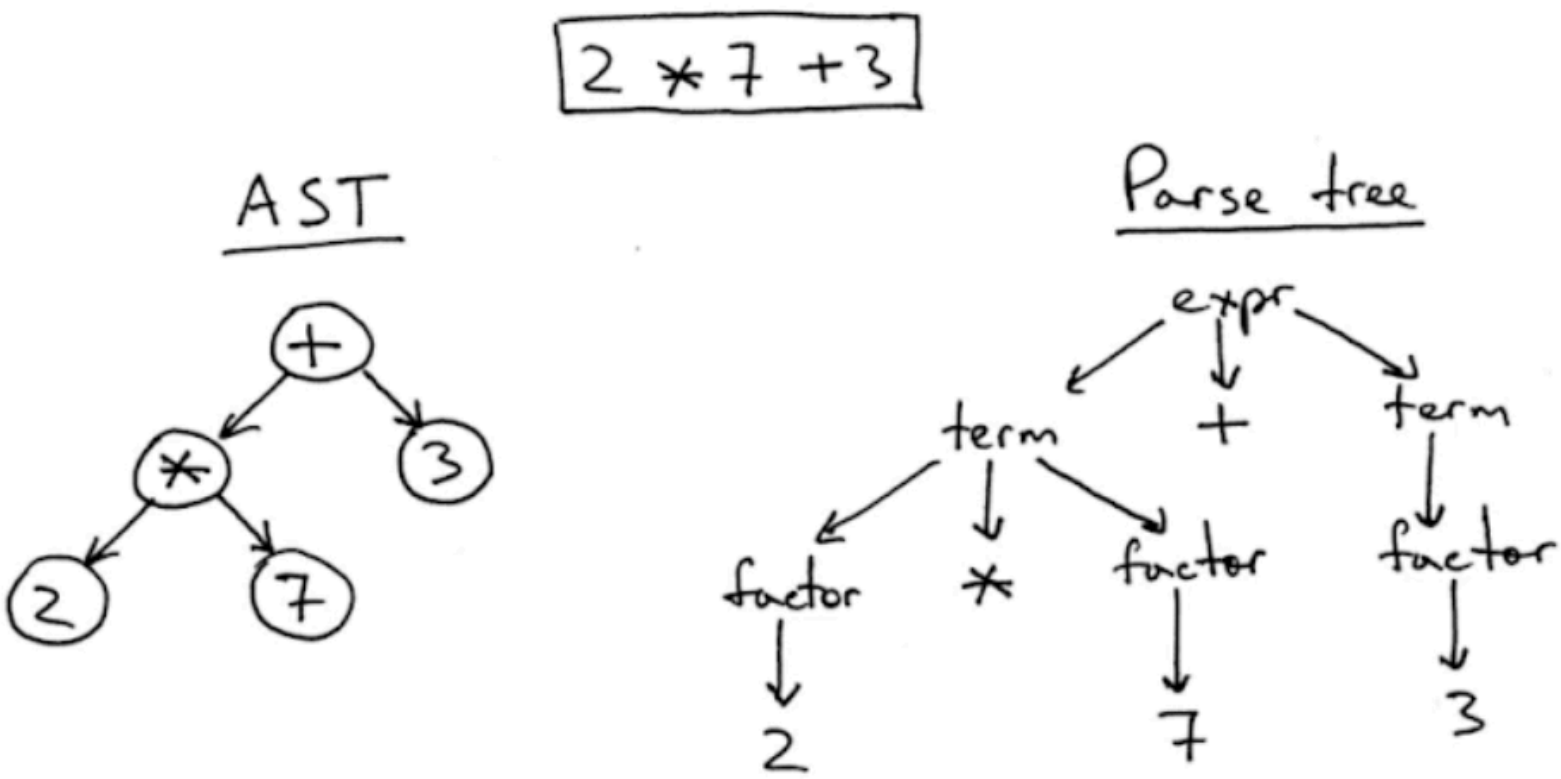


Данный скрипт первоначально «скармливается» лексеру (Lexer), он же Токенайзер. Мы преобразуем код в семантическое представление в виде токенов (набора маленьких атомарных единиц, которые имеют семантический смысл в контексте конкретного языка).

Если вы когда-то сталкивались в Python или других языках с ошибкой синтаксиса (SyntaxError), то эта ошибка появляется либо на этом этапе, либо на следующем при попытке понять корректна ли семантика программы. Далее, проходится синтаксический анализатор, у которого основная задача – составить дерево. Оно называется «**Abstract Syntax Tree**» (**AST**).

В семантике языка задан определенный порядок операций. В результате процессинга токенов анализатор составляет данное абстрактное синтаксическое дерево и его обход. В листьях дерева у нас находятся **операнды**, а в промежуточных вершинах – **операторы**.

Когда мы отходим от этого дерева, мы получаем процесс выполнения. В данном примере показан вариант математического выражения:



В языках программирования дерево будет немного сложнее, но концепция будет той же.

Ассемблер или байткод

На основе порядка операций по дереву, о котором мы говорили ранее, идет генерация набора команд/кодов. Они называются «Upcode» или «Byte-code» для конкретной архитектуры. Если бы мы говорили про компилируемые языки (например, Си), то на этом уровне была бы компиляция для аппаратной архитектуры нашего компьютера. В случае с языками виртуальной машины (Python, Java и т.д.) результатом компиляции будет представление в виде byte-code для интерпретаторов (виртуальной машины).

Что будет, если мы включим многопоточность?

Говоря о стандартном CPython, мы можем сказать, что сам интерпретатор написан на языке Си. Для выполнения каждой из данных инструкций внутри может запускаться большое количество инструкций на языке Си:

```
  0 LOAD_CONST               1 (2)
  3 LOAD_CONST               1 (2)
  6 BINARY_MULTIPLY
  7 PRINT_ITEM
  8 PRINT_NEWLINE
  9 LOAD_CONST               0 (None)
 12 RETURN_VALUE
```

Если операционная система может в любой момент переключить контекст, то это может произойти на уровне Си. Это может понести за собой серию ошибок на уровне интерпретатора.

Решение: запретить возможность перемещения контекста внутри byte-code. Byte-code не сможет выполняться в параллели, при этом мы минимизируем возникновение различных ошибок внутри интерпретатора из-за переключения контекста.

GIL (Global Interpreter Lock) запрещает более чем одному потоку что-либо выполнять, это положительно сказывается на операционной системе.

Для более подробного изучения предлагаем рассмотреть визуализацию от Дэвида Бизли по следующей ссылке.

Как вам урок?

Далее