Title

Введение:

Сейчас многие отрасли испытывают дефицит it-специалистов. Это открывает множество возможностей для людей для изучения программирования. Для этого в качестве первого языка многие выбирают именно Python3 из-за его изначальной простоты в освоении, широкой поддержки разработчиками и сообществом и востребованности во многих направлениях разработки ПО. Однако, на мой взгляд при изучении любого языка программирования кроме синтаксиса и умения работать с прикладными библиотеками необходимо также знание базовых (внутренних) принципов работы выбранного вами инструмента. В случае с Python3 речь идёт об интерпретаторе кода.

Цель: подготовить актуальный и структурированный доклад о принципе работы интерпретатора языка программирования Python3.

Задачи:

* Найти информацию об устройстве интерпретатора и его отличиях от компилятора.
* Изучить этапы исполнения программы (сценария) на языке Python3
* Продемонстрировать процесс исполнения программы на реальном примере и сделать выводы

Глава 1. Python - что такое интерпретатор и его отличия от компилятора.

Следует начать с того, что Python — мультипарадигменный высокоуровневый интерпретируемый язык программирования. Он был разработан в 1991 году Гвидо ван Россумом. Язык испытал влияние C и Lisp, что заметно из его синтаксиса и реализации поддержки функционального стиля программирования. Программа на языке Python может рассматриваться ещё и как сценарий — кода, решающего конкретную задачу «здесь и сейчас» (благодаря чему язык нашёл широкое применение в сфере автоматизации и тестирования ПО). Этот язык обладает внушительной базой кода — сейчас сайт pypi.org насчитывает более 400 тысяч библиотек.

Как было упомянуто ранее, Python является интерпретируемым языком. Интерпретатор — это программа, выполняющая интерпретацию. Данный процесс включает в себя построчный анализ, обработку и исполнение исходного кода. Также существуют компилируемые языки, такие как C, Java, Go и другие. Компилятор — программа, реализующая в целом тот же функционал, что и интерпретатор, однако вся обработка производится статически. Это значит, что все необходимые приведения типов и объем памяти для выполнения программы известны ещё до её выполнения. Таким образом, компилятор выявляет все возможные ошибки при потенциальном исполнении программы и требует их исправить до запуска. Интерпретатор же проверяет лишь синтаксическую правильность написанного кода и в таком случае позволит его запустить. Принято считать, что в своём составе интерпретатор также содержит компилятор, который отвечает за отсутствие синтаксических ошибок в исполняемой программе, при этом не проверяя корректность операций связанных с типами данных и необходимый объём памяти. Основное преимущество компилируемого подхода — заранее известные условия выполнения программы, а интерпретируемого — отсутствие необходимости тратить время на проверку кода, особенно в крупных проектах с большой базой исходного кода.

Глава 2. Процесс компиляции исходного кода.

Как было упомянуто ранее, процесс обработки кода на языке Python включает в себя компиляцию и интерпретацию на виртуальной машине Python Virtual Machine (PVM). Это показано на рисунке 1.

На протяжении всего процесса рассмотрения будет использоваться реализация CPython — каноничная реализация интерпретатора на языке C. По сути, любая программа на Python в процессе компиляции так или иначе превращается в программу на языке C. Подробнее процесс компиляции показан на рисунке 2. Как видим, процесс можно разбить на следущие этапы:

1. Создание синтаксического дерева (Parse Tree)
2. Генерация абстрактного синтаксического дерева (AST)
3. Генерация байт-кода
4. Оптимизация байт-кода
5. Генерация объектов кода

Для наглядности будет использована программа, содержащая в себе функцию, принимающую один числовой аргумент и увеличивающий его на единицу, затем принимающая с консоли число и выводящая результат выполнения функции на нём (рисунок 3).

Первый этап обработки программы — создание синтаксического дерева

Приложение

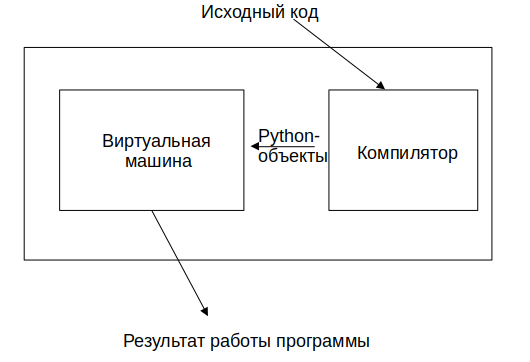
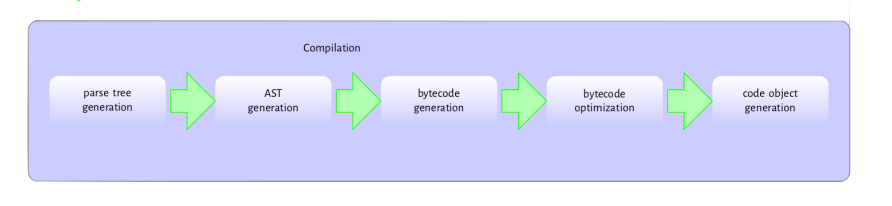


Рис. 1 Процесс обработки Python-кода

Рис. 2 Компиляция Python-кода

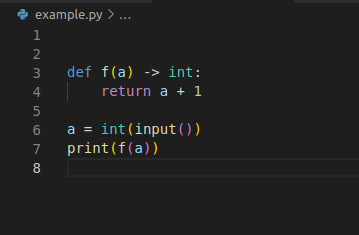


Рис.3 Используемая программа (example.py)