**ПОЧЕМУ PYTHON НЕ ЯВЛЯЕТСЯ ЯЗЫКОМ ПРОГРАММИРОВАНИЯ БУДУЮЩЕГО**

***Тулюгалиева Амина Ардаковна***

*студент, Алматинский технологический университет,*

*Казахстан, г. Алматы*

**АННОТАЦИЯ**

Цель данной статьи понять, почему все таки язык программирования Python устарел? Хотя на нем написано огромное количество мировых сайтов, приложений и тд. Учитывая повсеместную популярность Python на данный момент, несомненно, потребуется полдесятилетия, может быть, даже целое, чтобы любой из этих новых языков заменил его.

Какой из языков он будет - Rust, Go, Julia или новый язык будущего - трудно сказать на данном этапе. Но учитывая проблемы с производительностью, которые являются фундаментальными в архитектуре Python, вы неизбежно займёте его место. Первоначально Python был динамически исследован. Это означает, что для оценки выражения компилятор сначала ищет текущий блок, а затем последовательно все вызывающие функции.

Проблема с динамическим охватом заключается в том, что каждое выражение должно проверяться в каждом возможном контексте, что утомительно. Именно поэтому большинство современных языков программирования используют статическое сканирование.

Питон пытался перейти на статическое сканирование, но всё испортил. Обычно внутренние области - например, функции внутри функций - могли бы видеть и изменять внешние области. В Python внутренние области могут видеть только внешние области, но не изменять их. Это приводит к путанице.

**Ключевые слова:** Python, разработка программного обеспечения, инструменты разработки.

Программистскому сообществу потребовалось несколько десятилетий, чтобы оценить Python. Но с начала 2010-х годов популярность Java и JavaScript стремительно росла и в конечном итоге превысила C, C#, Java и JavaScript.

Но до каких пор эта тенденция будет продолжаться? Когда Python в конечном итоге будет заменен на другие языки, и почему?

Установить точную дату истечения срока действия Python было бы так много предположений, что это могло бы также пройти как научная фантастика. Вместо этого, я оценю достоинства, которые повышают популярность Python прямо сейчас, и слабые стороны, которые сломают его в будущем.

Что делает Python популярным прямо сейчас

Успех Python отражается в тенденциях переполнения стека, которые измеряют количество тегов в сообщениях на платформе. Учитывая размер StackOverflow, это довольно хороший показатель популярности языка.

**Что делает Python популярным прямо сейчас?**

Успех Python отражается в тенденциях переполнения стека, которые измеряют количество тегов в сообщениях на платформе. Учитывая размер StackOverflow, это довольно хороший показатель популярности языка.

 ***График 1.Теги различных языков программирования*** ***на StackOverflow***

В то время как язык программирования R в течение последних нескольких лет находится в состоянии застоя, а многие другие языки находятся в упадке, рост Python кажется непреодолимым. Почти 14% всех вопросов StackOverflow имеют метки "python", и тенденция растет. И есть несколько причин для этого.

**Устарел ли он?**

Python существует с 1990-х годов. Это не только означает, что у него было много времени, чтобы расти. Она также приобрела большое и поддерживающее сообщество.

Так что если у вас есть какие-либо проблемы во время кодирования на Python, высоки шансы, что вы сможете решить их с помощью одного поиска Google. Просто потому, что кто-то уже столкнулся с вашей проблемой и написал что-то полезное об этом.

**Это подходит только для начинающих**

Дело не только в том, что она существует десятилетиями, давая программистам время делать блестящие учебники. Более того, синтаксис Python хорошо читается людьми.

Для начала нет необходимости указывать тип данных. Вы просто объявляете переменную; Python поймет из контекста, является ли она целым числом, значением с плавающей точкой, булевым или чем-то еще. Это огромное преимущество для новичков. Если вам когда-либо приходилось программировать в С++, вы знаете, как это неприятно, что ваша программа не будет компилироваться, потому что вы поменяли поплавок на целое число.

И если вам когда-нибудь приходилось читать код Python и C++ бок о бок, вы будете знать, насколько понятен Python. Несмотря на то, что С++ был разработан с учетом английского языка, это довольно ухабистая чтение по сравнению с кодом Python.

**Насколько он разносторонний ?**

Поскольку Python существует так долго, разработчики создали пакет для всех целей. В наши дни вы можете найти пакет практически для всего.

Хотите работать с числами, векторами и матрицами? NumPy - ваш парень.

Хотите делать вычисления для техники и инженерии? Используйте SciPy.

Хотите стать великим в обработке и анализе данных? Дайте Пандасу шанс.

Хотите начать с искусственного интеллекта? Почему бы не использовать Scikit-Learn.

Какую бы вычислительную задачу вы ни пытались выполнить, есть вероятность, что для неё существует пакет Python. Это позволяет Python оставаться на вершине последних разработок, что видно из всплеска машинного обучения за последние несколько лет.

 **Какие недостатки Python ?**

Основываясь на предыдущих разработках, вы могли бы представить, что Python останется на вершине sh\*t на долгие годы. Но, как и у любой технологии, у Пайтона есть свои слабости. Я пройдусь по самым важным недостаткам, один за другим, и оценю, смертельны они или нет.

**Быстродействие**

Python медленный. Типа, очень медленный. В среднем, вам потребуется примерно от 2-10 раз больше времени, чтобы выполнить задачу с Python, чем с любым другим языком.

Это объясняется различными причинами. Одним из них является динамическая типизация - помните, что вам не нужно указывать типы данных, как в других языках. Это означает, что нужно использовать много памяти, потому что программе нужно резервировать достаточно места для каждой переменной, которую она работает в любом случае. Использование большого количества памяти приводит к большому количеству вычислительного времени.

Другая причина заключается в том, что Python может выполнять только одну задачу за раз. Это является следствием гибких типов данных - Python должен убедиться, что каждая переменная имеет только один тип данных, и параллельные процессы могут это испортить.

Для сравнения, ваш средний веб-браузер может запускать дюжину различных потоков одновременно. И есть некоторые другие теории вокруг, тоже.

Но в конце концов, ни одна из проблем скорости не имеет значения. Компьютеры и серверы стали настолько дешевыми, что мы говорим о доле секунды.

 **Какова область применения?**

Первоначально Python был динамически исследован. Это означает, что для оценки выражения компилятор сначала ищет текущий блок, а затем последовательно все вызывающие функции.

Проблема с динамическим охватом заключается в том, что каждое выражение должно проверяться в каждом возможном контексте, что утомительно. Именно поэтому большинство современных языков программирования используют статическое сканирование.

Питон пытался перейти на статическое сканирование, но всё испортил. Обычно внутренние области - например, функции внутри функций - могли бы видеть и изменять внешние области. В Python внутренние области могут видеть только внешние области, но не изменять их. Это приводит к путанице.

Несмотря на всю гибкость в Python, использование Lambdas является довольно ограничительным. Lambdas может быть только выражением в Python, а не выражением.

С другой стороны, переменные объявления и утверждения всегда являются утверждениями. Это означает, что Lambdas не может использоваться для них.

Это различие между выражениями и высказываниями является довол Пробел

В Python вы используете пробелы и отступы для обозначения различных уровней кода. Это делает его оптически привлекательным и интуитивно понятным для понимания.

Другие языки, например С++, больше полагаются на скобки и точки с запятой. Хотя это может не быть визуально привлекательным и дружественным для начинающих, это делает код гораздо более удобным. Для больших проектов это гораздо полезнее.

Новые языки, такие как Haskell, решают эту проблему: они полагаются на пробелы, но предлагают альтернативный синтаксис для тех, кто хочет обойтись без него.

Разработка мобильных устройств

Поскольку мы наблюдаем сдвиг от настольного компьютера к смартфону, ясно, что нам нужны надежные языки для создания мобильного программного обеспечения.

Но не многие мобильные приложения разрабатываются с помощью Python. Это не означает, что это невозможно - для этого существует пакет Python под названием Kivy.

Но Пайтон не был создан с учетом мобильного. Так что, даже если это может дать приемлемые результаты для основных задач, ваш лучший выбор - использовать язык, который был создан для разработки мобильного приложения. Некоторые широко используемые фреймворки для мобильных устройств включают React Native, Flutter, Iconic и Cordova.

Для ясности, ноутбуки и настольные компьютеры должны быть вокруг в течение многих лет в будущем. Но так как мобильный трафик давно превзошел трафик настольных компьютеров, можно с уверенностью сказать, что изучения Python недостаточно, чтобы стать опытным разработчиком.

Ошибки выполнения

Скрипт Python не компилируется сначала, а затем выполняется. Вместо этого, он компилируется каждый раз, когда вы выполняете его, так что любая ошибка кодирования проявляется во время выполнения. Это приводит к низкой производительности, расходу времени и необходимости проведения большого количества тестов.

Например, много тестов.

Это хорошо для новичков, так как тестирование учит их много. Но для опытных разработчиков отладка сложной программы на Python делает их неправильными. Этот недостаток производительности является самым большим фактором, который устанавливает отметку времени на Python произвольным и не встречается в других языках. ***График 2.*** ***Снимок тегов на различных языках программирования на StackOverflow.***

**Список литературы:**

1. Вордерман, К. Программирование на **Python**.
2. Гуриков, С.Р. Основы алгоритмизации и программирования на Python. Учебное пособие. Гриф МО РФ / С.Р. Гуриков. - М.: Инфра-М, Форум, 2018.
3. Эрик, Мэтиз Изучаем Python. Программирование игр, визуализация данных, веб-приложения / Мэтиз Эрик. - М.: Питер, 2017.
4. Бриггс, Джейсон Python для детей. Самоучитель по программированию