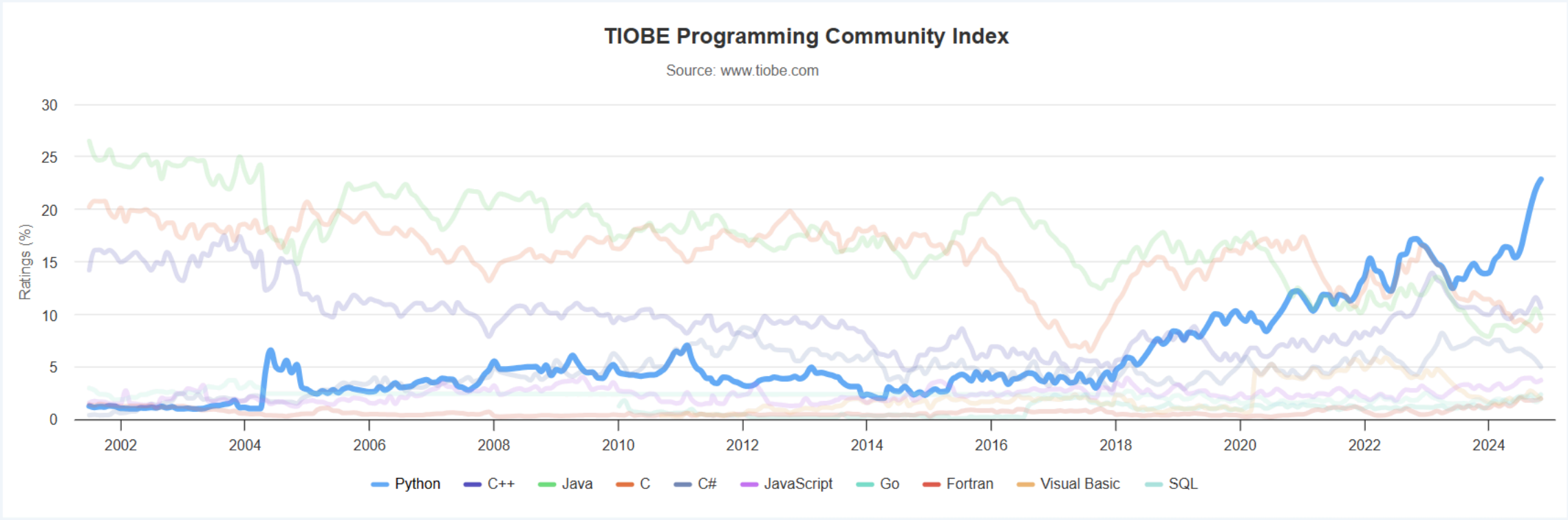
Другие языки программирования для анализа данных

Гейне М.А. (geine@bmstu.ru)

27.11.2024


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
Python
C++
Java
C
C#
JavaScript
Go
Fortran
Visual Basic
SQL

Давайте начнём с... мнений

SQL (Structured Query Language)

SQL - язык запросов, используемый в реляционных базах данных

- Необходим для манипулирования хранилищами данных и непосредственно данными
- SQLite, MySQL, PostgreSQL и множество других
- Не совсем язык программирования (хотя является тьюринг-полным)
- Возможно, лучше придерживаться ORM

Java

- Некогда самый популярный язык
- 56 billion devices globally
- Высокая производительность и совместимость
- Интегрируемость с Hadoop и Spark
- Тяжелый и неповоротливый, пора уже отпустить

```java
package springboot.topjava.ru;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/payment")
public class PaymentController {

    private final String sharedKey = "SHARED_KEY";

    private static final String SUCCESS_STATUS = "success";
    private static final String ERROR_STATUS = "error";
    private static final int CODE_SUCCESS = 100;
    private static final int AUTH_FAILURE = 102;

    @GetMapping
    public BaseResponse showStatus() {
        return new BaseResponse(SUCCESS_STATUS, 1);
    }

    @PostMapping("/pay")
    public BaseResponse pay(@RequestParam(value = "key") String key, @RequestBody PaymentRequest request) {

        final BaseResponse response;

        if (sharedKey.equalsIgnoreCase(key)) {
            int userId = request.getUserId();
            String itemId = request.getItemId();
            double discount = request.getDiscount();
            // Process the request
            // ....
            // Return success response to the client.
            response = new BaseResponse(SUCCESS_STATUS, CODE_SUCCESS);
        } else {
            response = new BaseResponse(ERROR_STATUS, AUTH_FAILURE);
        }
        return response;
    }
}
```

C/C++

- Сверхбыстрые и сверхэффективные
- Сложные в освоении
- Низкоуровневые
- "Макрокомпилятор ассемблера"
- Подходят скорее для разработки инструментов с большой нагрузкой

JavaScript

- JS - основа веба
- Кроссплатформенность с использованием Electron
- В основном используется для визуализаций и работы в реальном времени
- Есть средства для ML в браузере (TensorFlow.js, Pandas-js, Simple-statistics)
- Заметно менее производительный, чем Python

А теперь серьёзно

Julia

- Один из молодых языков (2009 год, но первая стабильная версия языка - 2018 год)
- Цель: высокая производительность C и лёгкость использования Python
- Компилируемый язык
- Полностью open-source
- Упор на параллелизм
- Оптимизация вычислений (поддержка линейной алгебры, эффективная работа с действительными числами, продвинутые массивы)
- Не поддерживает классы с включенными методами, а опирается на структуры и функции для них
- Интегрируемость с друими языками

Производительность

- Julia делает большой упор на производительность
- Большой прирост даёт свой специфический компилятор
- JIT-компиляция с высокой степенью оптимизации
- Нет явной типизации в коде, но при этом имеется сильная система выведения типов
- Не смешивает в себе два языка (как Python и C); высоко- и низко-уровневые функции сосуществуют
- Нет GIL :)
- Хорошая оптимизация памяти

Параллелизм "из коробки"

- Julia проектировалась с упором на современные процессоры и высоконагруженные системы
- Многопоточная обработка в циклах с использованием директивы `@threads`
- Распределённая обработка между несколькими процессами, которые потенциально запущены на разных машинах с директивой `@distributed`
- Поддержка корутин и обработки на GPU

```
using Base.Threads

function threaded_sum(A)
    s = 0.0
    @threads for i in 1:length(A)
        s += A[i]
    end
    return s
end
```

```julia
using Distributed
addprocs(4)  # Add 4 worker processes

@everywhere function compute(x)
    x^2
end

results = pmap(compute, 1:10)
```

```
using Distributed, DistributedArrays

A = distribute(rand(100, 100))  # Distribute a large array
B = A .* 2  # Element-wise multiplication
```

Оптимизированные массивы

- Массивы в Julia отдают приоритет столбцам
- Нативная поддержка векторизации и SIMD-операций
- Оптимизированная арифметика над массивами
- Интерфейс `AbstractArray` позволяет создавать собственые массивы, которые также смогут воспользоваться встроенными оптимизациями
- Набор специфических массивов для различных задач:
  - `SparceArrays` для разреженных матриц
  - `StaticArrays` для массивов фиксированной длины
  - `DistributedArrays` для распределённых между процессами массивов

Это всё ещё молодой язык

- Низкая популярность
- Малое сообщество
- Мало инструментов
- И т.д.
- Однако: инструментов мало, но они современные
- PlutoUI

R

- Один из наиболее широко используемых языков в DS
- Делает упор на статистические вычисления и визуализацию
- Первая версия в 1993 году
- Интерпретируемый язык, но с возможностью JIT-компиляции отдельных элементов
- Большое сообщество с сильной *академической* поддержкой
- Интегрируется с другими средствами посредством специфических интерфейсов, а также может быть интегрирован в другие среды (к примеру, Excel или PowerBI)

Обработка данных

- В R встроена возможность чтения структурированных данных, их просмотра и анализа
- Есть возможность манипуляции данными, фильрации и изменения формы набора данных
- `tidyverse` - коллекция дополнительных пакетов для обработки данных, составленная сообществом
  - `dplyr` для преобразования данных в декларативном синтаксисе
  - `tidyr` для очистки и форматирования наборов
  - `data.table` для работы с большими наборами

Статистический анализ

- Встроенные функции для дескриптивного анализа (подсчёт средних значений, стандартного отклонения и т.д.) и проверки гипотез
- Позволяет построить линейные и нелинейные модели на данных
- Анализ временных рядов, многомерный анализ, Байесовское моделирование

Визуализация данных

- R имеет встроенные средства построения графиков, по типу `plot` и `hist`
- `ggplot2` предоставляет более сложные возможности визуализации на основе Grammar of Graphics и послойного рисования
  - Grammar of Graphics - концепция разбиения графика на серию компонент, которые вместе составляют сложные визуализации
  - Фактически, те же средства, что в `matplotlib`
- Возможность создавать интерактивные визуализации: `plotly` улучшает `ggplot2`, а `shiny` позволит создать веб-приложение для изучения данных

Синтаксис

```r
x <- 5 # Assign value 5 to variable x

if (x > 10) {
  print("x is greater than 10")
} else {
  print("x is 10 or less")
}

result <- ifelse(x > 10, "x is greater than 10", "x is 10 or less")

my_function <- function(x, y) {
  return(x + y)
}
result <- my_function(5, 3)  # Calls the function with arguments 5 and 3

add_two <- function(x) {
  x + 2
}

v <- c(1, 2, 3, 4, 5) # vector
v[2]   # Accessing the second element (output: 2)
```

```
install.packages("Dict")

library(Dict)
ages <- Dict$new(
  Charlie = 40L,
  Alice = 30L,
  Bob = 25L,
  .class = "integer",
  .overwrite = TRUE
)

ages["Bob"]
ages$get("Bob")
ages$get(3L)

ages$sort()

ages$keys
ages$values
ages$items
ages$length
```

Это уже не молодой язык

- У R большое сообщество и много сложных, продвинутых инструментов
- Базовые инструменты позволяют быстро начать работать с данными, однако без сторонних модулей возможности быстро заканчиваются
- Синтаксис языка быстро становится запутанным
- Ощущается как очень нишевый

Scala

- Сокращение от Scalable Language
- Функциональный язык, работающий в JVM
- Первый публичный релиз в 2004 году
- Статическая типизация с системой вывода типов
- Совместима со всем набором средств Java
- Делает акцент на неизменяемости, что облегчает распараллеливание

В разрезе Data Science

- Высокая производительность: статическая типизация, оптимизация в JVM, неизменяемость и ленивые вычисления
- Распределённые и параллельные вычисления поддерживаются "из коробки"
- Нацеленность на обработку больших данных и связь с Apache Spark
- Type Safety позволяет организовать большие пайплайны обработки данных, выявляя проблемы на ранних стадиях

Однако есть и недостатки

- Высокий порог вхождения
- Малая экосистема для Data Science
- Долгий цикл разработки в сравнении с другими языками (проблемы Java)
- Нет собственных высокоуровневых средств визуализации

Этот язык должен спасти ~~человечество~~ JVM

- Язык ощущается всё ещё молодым и активно развивающимся
- На нём работают крупные продукты для работы с большими данными
- Сильная функциональная парадигма без отказа от ООП
- Сохраняет и приумножает наследие Java
- Конечно, с ним не сделать отчёт о данных, но он позволит организовать эффективную обработку данных

```scala
package var3.task1

import scala.util.{Try, Using}
import java.io.PrintWriter

// Reads words from a file, flattening the lines into a single list of words
def readFromFile(filename: String): Try[List[String]] =
  Using(scala.io.Source.fromFile(filename)) { source =>
    source.getLines().flatMap(_.split("\\s+")).toList
  }

// Finds all words where the last character matches the first character of the next word
def findMatchingWords(words: List[String]): List[String] = {
  words.sliding(2).collect {
    case List(first, second) if first.last == second.head => first
  }.toList
}

// Reads input, processes words, and writes output to a file
def processFile(input: String, output: String): Unit = {
  val result = for {
    words <- readFromFile(input)
    matchingWords = findMatchingWords(words)
    _ <- Using(new PrintWriter(output)) { writer =>
      matchingWords.foreach(writer.println)
    }
  } yield ()

  result.recover {
    case e: Exception => println(s"An error occurred: ${e.getMessage}")
  }
}
```

```java
import java.io.*;
import java.nio.file.*;
import java.util.*;
import java.util.stream.*;

public class WordProcessor {

    // Reads words from a file, flattening the lines into a single list of words
    public static List<String> readFromFile(String filename) throws IOException {
        return Files.lines(Paths.get(filename))
                    .flatMap(line -> Arrays.stream(line.split("\\s+")))
                    .collect(Collectors.toList());
    }

    // Finds all words where the last character matches the first character of the next word
    public static List<String> findMatchingWords(List<String> words) {
        List<String> result = new ArrayList<>();
        for (int i = 0; i < words.size() - 1; i++) {
            String first = words.get(i);
            String second = words.get(i + 1);
            if (first.charAt(first.length() - 1) == second.charAt(0)) {
                result.add(first);
            }
        }
        return result;
    }

    // Writes a list of words to a file
    public static void writeToFile(String filename, List<String> words) throws IOException {
        try (PrintWriter writer = new PrintWriter(new FileWriter(filename))) {
            for (String word : words) {
                writer.println(word);
            }
        }
    }

    // Main process function to integrate reading, processing, and writing
    public static void processFile(String input, String output) {
        try {
            List<String> words = readFromFile(input);
            List<String> matchingWords = findMatchingWords(words);
            writeToFile(output, matchingWords);
        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
        }
    }

    // Main method for testing
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java WordProcessor <inputFile> <outputFile>");
            return;
        }
        processFile(args[0], args[1]);
    }
}
```

Обратите внимание

- Во многих рассмотренных языках и инструментах если не преобладает, то хорошо поддерживается функциональный стиль
- Эффективная обработка данных зависит от хорошей параллельности
- Хорошая параллельность зависит от изолируемости обработки
- Изолируемость обработки легко достижима при неизменяемости данных
- Хорошая параллельность выигрывает от возможности организовать цепь обработки данных
- Неизменяемость и склонность к цепям обработки - качества функционального программирования

Спасибо за внимание

(∩ ` —´) ⊃ ━☆ﾟ.*･｡ﾟ λ