Zurück zur Universität

Eintauchen in die wissenschaftliche Programmierung

Willem L. Middelkoop
Feb. 8, 2023

Diesen Monat ging es für mich zurück an die Uni, für einen speziellen Kurs über wissenschaftliches Programmieren. Mit den zahlreichen Fortschritten im Bereich Machine Learning (in PowerPoint-Sprache "KI" genannt) der letzten Zeit, einschließlich ChatGPT, schien es der ideale Zeitpunkt zu sein, um in die Python-Programmierung einzutauchen. Begleiten Sie mich auf dieser Reise.

Warum Python für wissenschaftliches Programmieren?

Python hat in der wissenschaftlichen Gemeinschaft immense Popularität erlangt, und das aus gutem Grund. Seine Einfachheit und Lesbarkeit machen es für Neueinsteiger zugänglich, während es gleichzeitig die Leistungsfähigkeit behält, komplexe Aufgaben zu bewältigen. Die Vielseitigkeit der Sprache ermöglicht es Forschern und Wissenschaftlern, Algorithmen und Modelle relativ einfach zu implementieren.

Eine umfangreiche Auswahl an Bibliotheken, die speziell für wissenschaftliches Rechnen entwickelt wurden, wie NumPy, SciPy, Pandas und Matplotlib, stärkt Pythons Position in diesem Bereich. Diese Bibliotheken bieten umfassende Funktionen und Werkzeuge für Datenanalyse, statistische Modellierung und Visualisierung und optimieren so den wissenschaftlichen Programmierprozess.

Im Vergleich zu anderen Programmiersprachen wie JavaScript oder Ruby, die häufig in nicht-wissenschaftlichen Kontexten wie der Webentwicklung eingesetzt werden, verfügt Python über eine einfachere Syntax. Diese Eigenschaft ermöglicht es Benutzern, sich auf das vorliegende Problem zu konzentrieren, anstatt mit sprachlichen Feinheiten zu kämpfen. Darüber hinaus macht Pythons umfangreiches wissenschaftliches Ökosystem es zu einer geeigneteren Wahl für Forschungsanwendungen.

Betrachten Sie die folgenden zwei Codebeispiele, eines in C++ und das andere in Python 3. Jedes Programm führt identische Aufgaben aus: Es fragt den Benutzer nach seinem Namen und Geburtsdatum, führt eine grundlegende Eingabevalidierung durch und gibt schließlich das aktuelle Alter des Benutzers aus.

```cpp
#include <iostream>
#include <string>
#include <sstream>
#include <ctime>

bool validate_date(const std::string &date_string, int &day, int &month, int &year) {
    std::istringstream iss(date_string);
    char delimiter;
    if (iss >> day >> delimiter >> month >> delimiter >> year) {
        return true;
    }
    return false;
}

int calculate_age(int day, int month, int year) {
    time_t now = time(0);
    tm *ltm = localtime(&now);

    int current_year = 1900 + ltm->tm_year;
    int current_month = 1 + ltm->tm_mon;
    int current_day = ltm->tm_mday;

    int age = current_year - year - ((current_month < month) || (current_month == month && current_day
< day));
    return age;
}

int main() {
    std::string name, dob_string;
    int day, month, year;

    std::cout << "Please enter your name: ";
    std::getline(std::cin, name);

    do {
        std::cout << "Please enter your date of birth (dd-mm-yyyy): ";
        std::getline(std::cin, dob_string);

        if (!validate_date(dob_string, day, month, year)) {
            std::cout << "Invalid date format. Please try again." << std::endl;
        }
    } while (!validate_date(dob_string, day, month, year));

    int age = calculate_age(day, month, year);
    std::cout << name << ", your age is " << age << " years." << std::endl;

    return 0;
}
```

C++-Beispielcode: Bestimmung des Alters des Benutzers

```python
from datetime import datetime

def get_user_input(prompt):
    while True:
        user_input = input(prompt).strip()
        if user_input:
            return user_input

def validate_date(date_string):
    try:
        date_obj = datetime.strptime(date_string, '%d-%m-%Y')
        return date_obj
    except ValueError:
        return None

def calculate_age(dob):
    today = datetime.now()
    age = today.year - dob.year - ((today.month, today.day) < (dob.month,
dob.day))
    return age

name = get_user_input("Please enter your name: ")
dob_string = ""

while not dob_string:
    dob_string = get_user_input("Please enter your date of birth (dd-mm-yyyy): ")
    dob = validate_date(dob_string)
    if not dob:
        dob_string = ""
        print("Invalid date format. Please try again.")

age = calculate_age(dob)
print(f"{name}, your age is {age} years.")
```

Python 3-Beispielcode: Bestimmung des Alters des Benutzers

Man kann leicht die gegensätzliche Syntax der beiden Sprachen beobachten, wobei Python der Alltagssprache näher kommt. Dieser Aspekt von Python verbessert oft die Lesbarkeit und das Verständnis für viele Benutzer.

Ein weiterer Faktor, der zum Erfolg von Python im wissenschaftlichen Programmieren beiträgt, ist seine florierende Community. Forscher und Entwickler aus verschiedenen Disziplinen teilen Wissen, bieten Unterstützung und tragen zu Open-Source-Projekten bei, wodurch eine Umgebung der Zusammenarbeit und des kontinuierlichen Lernens gefördert wird.

Der Kurs

Im Laufe des Kurses wurden die absoluten Grundlagen des Programmierens durch die Bearbeitung von Problemen aus verschiedenen wissenschaftlichen Bereichen erforscht. Nach Abschluss wurde ein solides Verständnis der Programmierprinzipien erworben, mit der Fähigkeit, diese in verschiedenen Bereichen und Projekten anzuwenden.

Die Universiteit van Amsterdam auf dem Science Park Campus - Freunde des Blogs werden mein Fahrrad erkennen

Die Wiederholung der Grundlagen des Programmierens, selbst für erfahrene Programmierer (wie mich selbst), kann sich als immens nützlich erweisen. Im Laufe der Zeit ist es nicht ungewöhnlich, dass Programmierer Gewohnheiten oder Abkürzungen entwickeln, die möglicherweise nicht den Best Practices entsprechen. Durch die Rückkehr zu den Grundlagen können Sie schädliche Gewohnheiten verlernen und ein starkes Fundament in den Programmierprinzipien wiederherstellen. Darüber hinaus bedeutet die schnelllebige Natur der Tech-Industrie, dass ständig Innovationen und Updates auftauchen. Durch die Wiederholung der Grundlagen können erfahrene Programmierer mit den neuesten Entwicklungen und Methoden Schritt halten. Dieser Prozess der Neubewertung bietet auch die Möglichkeit, das Programmieren mit einem unvoreingenommenen Blick anzugehen.

[image]

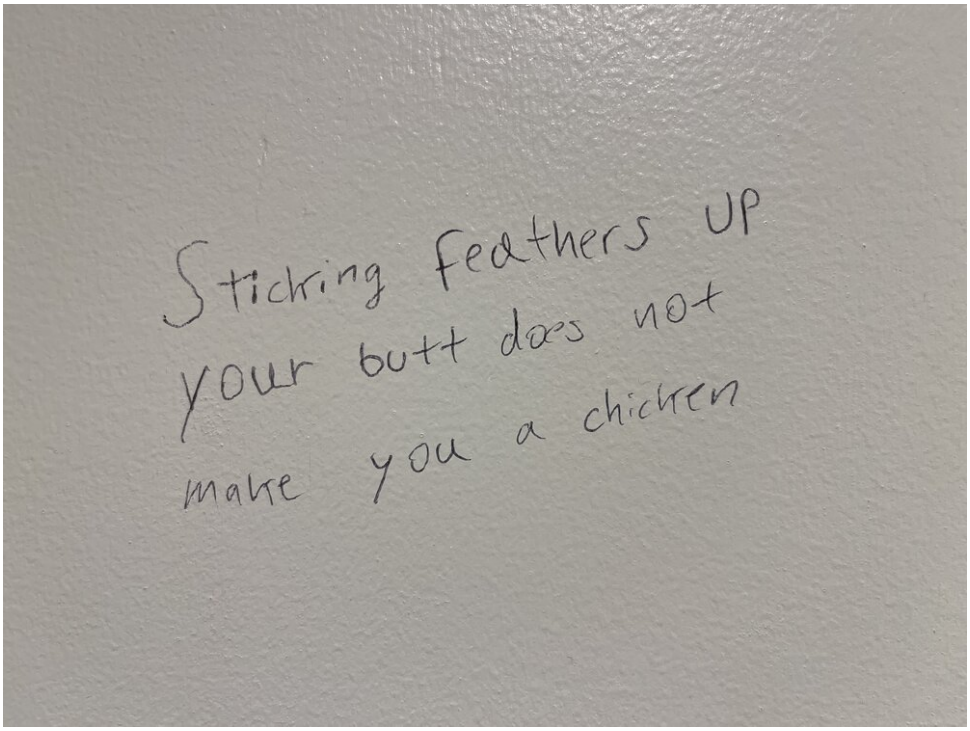
Der Kurs bestand aus mehreren Modulen, beginnend mit Level 1, wo eine Wahl zwischen ALGORITHMEN, die sich auf die Zerlegung intuitiver Probleme in Schritte konzentrierten, die ein Computer verstehen konnte, und ZAHLEN, einem mathematisch orientierten Modul, das sich mit Zahleneigenschaften befasste, ohne vorherige mathematische Kenntnisse zu erfordern, getroffen wurde. In Level 2 bestand die Wahl zwischen TEXT, der sich auf die Verarbeitung natürlicher Sprache und die Stimmungsanalyse von Tweets konzentrierte, und NUMERISCHE INTEGRATION, wo wichtige Techniken zur Bestimmung von Oberflächen unter Funktionen vermittelt wurden, wenn die traditionelle Integration nicht ausreichte. Level 3 umfasste die Arbeit mit BIG-DATA, die Analyse großer Datensätze und Wettermuster in den Niederlanden. Ein optionales Bonuslevel, BEWEGUNG, bot eine Simulation des Grabens eines Tunnels durch den Planeten und erforschte physikalische Probleme, die mit Computerunterstützung lösbar sind.

Meine Teilnahme an einem Programmierkurs an der Universität (Kommilitonen aus Datenschutzgründen unkenntlich gemacht)

[image]

Schlussfolgerung

Egal wie alt oder weise man ist, es gibt immer Raum, etwas Neues zu lernen, insbesondere mit den Fortschritten im maschinellen Lernen wie GPT4. Darüber hinaus bietet die Rückkehr zur Universität eine erfrischende Abwechslung und einen völlig anderen Kontext im Vergleich zum kommerziellen oder beruflichen Umfeld, in dem ich normalerweise tätig bin. Machen Sie sich das Mantra zu eigen: Bleiben Sie hungrig, bleiben Sie verrückt!



Die wöchentlichen Laborsitzungen bieten eine wunderbare Gelegenheit, bei Ihren Programmierabenteuern auf dem Laufenden zu bleiben und sicherzustellen, dass Sie Hilfe von anderen erhalten, wenn Sie auf ein Problem stoßen.