De vuelta a la Universidad

Inmersión en la Programación Científica

Willem L. Middelkoop
Feb. 8, 2023

Este mes, regresé a la universidad para un curso especial sobre programación científica. Con numerosos avances recientes en el aprendizaje automático (referido como "IA" en la jerga de PowerPoint), incluyendo ChatGPT, sentí que era el momento ideal para profundizar en la programación con Python. Acompáñenme en este viaje.

¿Por qué Python para la programación científica?

Python ha ganado una inmensa popularidad dentro de la comunidad científica, y por una buena razón. Su simplicidad y legibilidad lo hacen accesible para los recién llegados, al tiempo que conserva la potencia para abordar tareas complejas. La versatilidad del lenguaje permite a los investigadores y científicos implementar algoritmos y modelos con relativa facilidad.

Una amplia gama de bibliotecas diseñadas para la computación científica, como NumPy, SciPy, Pandas y Matplotlib, refuerza la posición de Python en este dominio. Estas bibliotecas ofrecen funciones y herramientas completas para el análisis de datos, el modelado estadístico y la visualización, lo que agiliza el proceso de programación científica.

En comparación con otros lenguajes de programación, como JavaScript o Ruby, que a menudo se utilizan en contextos no científicos como el desarrollo web, Python cuenta con una sintaxis más sencilla. Esta característica permite a los usuarios centrarse en el problema en cuestión en lugar de luchar con las complejidades del lenguaje. Además, el extenso ecosistema científico de Python lo convierte en una opción más adecuada para aplicaciones de investigación.

Examine los dos siguientes ejemplos de código, uno en C++ y el otro en Python 3. Cada programa realiza tareas idénticas: solicita el nombre y la fecha de nacimiento del usuario, realiza una validación básica de la entrada y, finalmente, revela la edad actual del usuario.

```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;
}
```

Código de ejemplo en C++: Determinando la edad del usuario

```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.")
```

Código de ejemplo en Python 3: Determinando la edad del usuario

Se puede observar fácilmente la sintaxis contrastante entre los dos lenguajes, con Python teniendo un mayor parecido al inglés cotidiano. Este aspecto de Python a menudo mejora la legibilidad y la comprensión para muchos usuarios.

Otro factor que contribuye al éxito de Python en la programación científica reside en su próspera comunidad. Investigadores y desarrolladores de diversas disciplinas comparten conocimientos, ofrecen apoyo y contribuyen a proyectos de código abierto, fomentando un entorno de colaboración y aprendizaje continuo.

El Curso

A lo largo del curso, se exploraron los fundamentos absolutos de la programación abordando problemas de diversos dominios científicos. Al finalizar, se adquirió una sólida comprensión de los principios de programación, con la capacidad de aplicarlos en diferentes campos y proyectos.

La Universidad de Ámsterdam en el campus Science Park - los amigos del blog reconocerán mi bicicleta

Repasar los fundamentos de la programación, incluso para programadores experimentados (como yo), puede resultar inmensamente beneficioso. Con el tiempo, no es raro que los programadores desarrollen hábitos o atajos que pueden no adherirse a las mejores prácticas. Al volver a los fundamentos, se pueden desaprender los hábitos perjudiciales y restablecer una base sólida en los principios de programación. Además, la naturaleza acelerada de la industria tecnológica significa que las innovaciones y actualizaciones surgen constantemente. Al repasar los fundamentos, los programadores experimentados pueden mantenerse al día con los últimos avances y metodologías. Este proceso de reevaluación también ofrece la oportunidad de abordar la programación con una pizarra limpia.

[image]

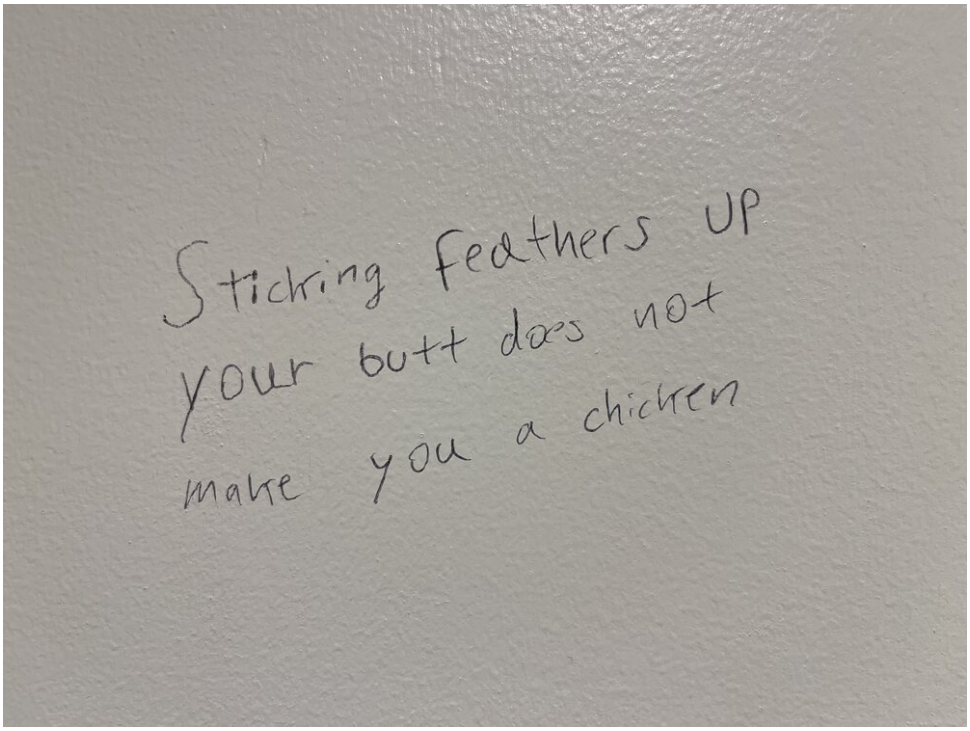
El curso constó de varios módulos, comenzando con el Nivel 1, donde se eligió entre ALGORITMOS, que se centró en descomponer problemas intuitivos en pasos que una computadora pudiera entender, y NÚMEROS, un módulo de orientación matemática que profundizó en las propiedades de los números sin requerir conocimientos matemáticos previos. En el Nivel 2, la elección fue entre TEXTO, que se centró en el procesamiento del lenguaje natural y el análisis de sentimiento de tweets, e INTEGRACIÓN NUMÉRICA, donde se enseñaron técnicas importantes para determinar áreas superficiales bajo funciones cuando la integración tradicional no era suficiente. El Nivel 3 implicó trabajar con GRANDES DATOS, analizando grandes conjuntos de datos y patrones climáticos en los Países Bajos. Un nivel de bonificación opcional, MOVIMIENTO, proporcionó una simulación de la excavación de un túnel a través del planeta y exploró problemas de física que se pueden resolver con asistencia informática.

Yo participando en una clase de programación en la Universidad (los compañeros de clase están difuminados por razones de privacidad)

[image]

Conclusión

No importa la edad o la sabiduría de uno, siempre hay espacio para aprender algo nuevo, particularmente con los avances en el aprendizaje automático como GPT4. Además, volver a la universidad proporciona un cambio refrescante, presentando un contexto completamente diferente en comparación con el ámbito comercial o profesional en el que normalmente opero. Adopta el mantra: ¡mantén el hambre, mantén la locura!



Las sesiones de laboratorio semanales ofrecen una deliciosa oportunidad para mantenerse al día con tus aventuras de programación, asegurándote de tener una mano amiga de otros siempre que te encuentres con un obstáculo.