Capítulo 9

Del algoritmo al plano: explorando la enseñanza de programación con C y Python en la ingeniería civil

María Elena Romero Gastelú
Janette Araceli Castellanos Barajas
Luis Ernesto Ayala Hernández

elena.romero@academicos.udg.mx
(Autor de correspondencia)

DOI: **https://doi.org/10.61728/AE26002101**

Resumen

En este capítulo se presenta el desarrollo de un análisis comparativo entre los lenguajes de programación C y Python, con el objetivo de identificar su pertinencia como herramientas para la enseñanza inicial de la programación a nivel ingeniería, empleando la metodología de Aprendizaje Basado en Proyectos (ABP) en la asignatura de Programación Aplicada a la Ingeniería, impartida en el primer semestre de la carrera de Ingeniería Civil del Centro Universitario de Ciencias Exactas e Ingenierías. La asignatura tiene como finalidad promover un aprendizaje activo mediante la resolución de problemáticas contextualizadas en escenarios reales de la ingeniería civil, favoreciendo la integración de saberes, el pensamiento crítico y el trabajo colaborativo.

La comparación consideró criterios como la curva de aprendizaje, la comprensión de la lógica computacional, la estructura y claridad sintáctica del lenguaje, así como la facilidad para implementar soluciones a problemas de ingeniería. Lo anterior como parte de esta experiencia didáctica, durante el ciclo escolar 2025-B.

El análisis comparativo tuvo como propósito central fortalecer en los estudiantes una comprensión significativa del proceso de aprendizaje de la programación, mediante el análisis de dos lenguajes de programación (C y Python) en diferentes grupos de estudio. Desde esta perspectiva, se consideró el enfoque ABP, ya que es un modelo de aprendizaje activo en el que los alumnos planifican, desarrollan y evalúan proyectos con aplicaciones prácticas que trascienden el aula, favoreciendo la construcción autónoma del conocimiento (Blank, 1997; Harwell, 1997; Martí, 2010).

Para llevar a cabo el análisis comparativo de los lenguajes de programación planteados en este proyecto, las actividades de la asignatura se estructuraron de manera progresiva mediante una metodología basada en proyectos, permitiendo que los estudiantes adquirieran las bases de la programación y al mismo tiempo aplicaran los conceptos computa-

cionales necesarios para la solución de problemas propios del ámbito de la ingeniería civil. La estrategia de implementación incluyó el diseño de actividades basadas en competencias, la elaboración de prácticas contextualizadas (problemas de ingeniería civil), el uso de recursos didácticos digitales y la evaluación continua a través de entregables (prácticas, exámenes parciales y un producto final integrador).

Los resultados evidencian un impacto positivo en los estudiantes, quienes manifestaron mayor interés y facilidad de aprendizaje en el uso del lenguaje de Python, gracias a su sintaxis simple y el uso de bibliotecas orientadas al área ingenieril. Además, el uso del enfoque ABP como una metodología favorece el aprendizaje significativo y contextualizado en los primeros semestres de la formación en ingeniería.

Palabras clave: *ABP, enseñanza de la programación, formación en ingeniería civil, innovación docente, lenguaje de programación C, lenguaje de programación Python.*

1. Introducción

En la actualidad, los avances acelerados en ciencia y tecnología demandan que las instituciones de educación superior fortalezcan el desarrollo de competencias digitales y pensamiento computacional desde los primeros semestres de las carreras de ingeniería. La alfabetización digital y la capacidad de resolver problemas mediante la programación se han convertido en habilidades fundamentales para el ejercicio profesional en contextos altamente tecnificados, por lo que su incorporación efectiva en la formación universitaria representa un reto clave para responder a las necesidades del entorno social y productivo (UNESCO, 2019; SEP, 2022).

En este contexto, la enseñanza de la programación en los primeros semestres de las ingenierías requiere estrategias didácticas que favorezcan no solo la adquisición de conocimientos técnicos, sino también el desarrollo del pensamiento lógico, la autonomía y la capacidad de aplicar lo aprendido en situaciones reales.

Existen estudios comparativos que han abordado las diferencias entre C y Python en términos de facilidad de uso y aprendizaje para princi-

piantes. Algunas investigaciones empíricas realizadas con estudiantes de cursos introductorios muestran que los participantes encuentran la sintaxis y estructura de Python significativamente más accesibles y fáciles de aprender que C, lo cual se traduce en mejores resultados de aprendizaje y comprensión en fases iniciales de la formación en programación (Asghar et al., 2025).

Lenguaje C

El lenguaje C ha sido históricamente uno de los pilares en la enseñanza de la programación en las ingenierías, debido a su enfoque estructurado y a su cercanía con el funcionamiento interno de los sistemas computacionales (Ritchie, 1993). Su aprendizaje favorece el desarrollo del pensamiento lógico, el análisis algorítmico y la comprensión de conceptos fundamentales, como el control del flujo y la gestión de memoria (Kernighan & Ritchie, 1988). Aunque presenta una curva de aprendizaje exigente, la literatura coincide en que su enseñanza en los primeros semestres contribuye a la formación de bases sólidas para la resolución de problemas propios del ámbito ingenieril (Sebesta, 2016).

Lenguaje Python

El lenguaje Python se ha consolidado como una herramienta ampliamente utilizada en la enseñanza inicial de la programación, gracias a su sintaxis clara y legible, que reduce la carga cognitiva de los estudiantes y facilita la comprensión de la lógica computacional (Van Rossum & Drake, 2011). Diversos estudios señalan que Python favorece un aprendizaje más rápido y motivador, permitiendo a los alumnos concentrarse en la resolución de problemas más que en aspectos sintácticos (Robins et al., 2003). En el contexto de la ingeniería, Python resulta especialmente pertinente para la introducción a la programación, el análisis de datos y la modelación, constituyéndose como un lenguaje accesible y eficaz en los primeros semestres de formación.

Aprendizaje basado en proyectos

El ABP es ampliamente utilizado como una estrategia pedagógica idónea para responder a las demandas actuales de la formación en ingeniería, al colocar al estudiante en el centro del proceso de aprendizaje. Según Harwell (1997), este enfoque se caracteriza por involucrar a los alumnos en la planeación, desarrollo y evaluación de proyectos con una aplicación directa en contextos reales. De manera complementaria, Barrows (1986) señala que el ABP parte del planteamiento de problemas auténticos como eje para la construcción e integración de nuevos conocimientos. Bajo este enfoque, el ABP favorece un aprendizaje activo, significativo y contextualizado, en consonancia con las competencias y propósitos formativos propios de las carreras de ingeniería.

Además, diversos estudios señalan que los enfoques tradicionales centrados exclusivamente en la transmisión de contenidos pueden resultar insuficientes para lograr aprendizajes significativos en esta área, lo que hace necesario el uso de metodologías activas centradas en el estudiante (Díaz Barriga, 2010; Coll, 2006).

Este capítulo aborda un estudio comparativo entre los lenguajes de programación C y Python, cuyo propósito es analizar su idoneidad como herramientas didácticas para la enseñanza introductoria de la programación en el ámbito de la ingeniería. Dicho análisis se desarrolla en el contexto de la asignatura Programación Aplicada a la Ingeniería, correspondiente al primer semestre de la carrera de Ingeniería Civil del Centro Universitario de Ciencias Exactas e Ingenierías.

La experiencia se sustenta en la aplicación de la metodología ABP, la cual permitió situar al estudiante como eje central del proceso formativo, promoviendo la participación, el razonamiento lógico y la resolución de problemas contextualizados en escenarios reales de la ingeniería. A través de esta metodología, los estudiantes de diferentes grupos sometidos a estudio tuvieron la oportunidad de aplicar los conceptos fundamentales de programación (C y Python) mediante el desarrollo de actividades prácticas, exámenes parciales y un proyecto integrador, lo que facilitó la observación del impacto de cada lenguaje en el proceso de aprendizaje.

El análisis comparativo consideró aspectos clave como la comprensión de la lógica computacional, la estructura y claridad sintáctica, la curva de aprendizaje y la facilidad para implementar soluciones a problemas de ingeniería, permitiendo reflexionar sobre las ventajas y limitaciones de cada lenguaje en los primeros semestres de la formación profesional. Los resultados obtenidos ofrecen elementos de análisis relevantes para la toma de decisiones pedagógicas en la enseñanza de la programación, así como para el diseño de estrategias didácticas que contribuyan a una formación más sólida, contextualizada y acorde con las herramientas tecnológicas informáticas (TI) actuales de las carreras de ingeniería.

Las tecnologías de la información han transformado de manera significativa la práctica de la ingeniería civil, tanto en el ámbito académico como profesional. Su aplicación ha permitido optimizar los procesos de planeación, análisis y diseño de proyectos mediante herramientas computacionales y modelos de simulación.

Asimismo, ha fortalecido el razonamiento lógico y la capacidad de resolución de problemas complejos a través del uso de lenguajes de programación y análisis de datos. La integración de tecnologías digitales ha favorecido la vinculación entre teoría y práctica, facilitando la toma de decisiones fundamentadas. De igual forma, ha mejorado la gestión y supervisión de proyectos en términos de tiempo, costo y calidad. Estos avances contribuyen a una formación más integral y actualizada del ingeniero civil. Por último, la incorporación de las TI ha incrementado la competitividad profesional y la adaptación a entornos tecnológicos en constante evolución.

2. Desarrollo de la práctica docente

La práctica docente implementada en la asignatura Programación Aplicada a la Ingeniería tuvo como propósito desarrollar un análisis comparativo entre los lenguajes de programación C y Python, con el fin de identificar su pertinencia como herramientas para la enseñanza inicial de la programación en el nivel de ingeniería. Para alcanzar este objetivo, la experiencia se diseñó e implementó bajo el enfoque ABP, orientado a fortalecer el aprendizaje activo y significativo de la programación en

estudiantes de primer semestre de la carrera de Ingeniería Civil. La implementación se llevó a cabo durante el ciclo escolar 2025-B, considerando la necesidad de articular los contenidos teóricos con actividades prácticas contextualizadas en problemáticas reales del ámbito de la ingeniería civil.

La planeación del curso contempló una estructura progresiva, organizada en unidades temáticas que abordaron los fundamentos de la programación, tales como el diseño de algoritmos, el uso de estructuras de control, la manipulación de datos y la resolución de problemas computacionales enfocados a la Ingeniería Civil. Estos contenidos se trabajaron de manera paralela en 5 grupos de estudio, de los cuales 2 estuvieron orientados al lenguaje de Python y los otros 3 al lenguaje C, lo que permitió analizar de forma sistemática el impacto de cada lenguaje en el proceso de aprendizaje, particularmente en la comprensión de la lógica computacional y la estructuración de soluciones algorítmicas.

Durante el desarrollo del curso, las sesiones se impartieron bajo un enfoque teórico-práctico, en el que el docente fungió como mediador del aprendizaje, promoviendo la participación activa, el trabajo colaborativo y la reflexión constante sobre la lógica de programación. Para cada tema se diseñaron actividades prácticas y tareas, orientadas a la resolución de problemas propios de la ingeniería civil, tales como cálculos básicos, análisis de datos y modelación de situaciones sencillas, permitiendo a los estudiantes aplicar de forma inmediata los conceptos revisados en clase y evidenciar las particularidades del lenguaje de programación empleado.

Como parte del proceso de evaluación continua, se aplicaron exámenes parciales mediante formularios digitales, los cuales permitieron valorar la comprensión de los contenidos, el desarrollo del razonamiento lógico y el dominio de la sintaxis y estructura de cada lenguaje de programación. Estos instrumentos facilitaron la identificación de avances y áreas de oportunidad, así como la retroalimentación oportuna, elementos clave para sustentar el análisis comparativo entre C y Python desde una perspectiva formativa.

El proyecto integrador, permitió a los estudiantes desarrollar una solución informática que incorporó los principales contenidos abordados durante el semestre. Al mismo tiempo, este proyecto permitió evaluar la capacidad de los alumnos para integrar conocimientos, estructurar

algoritmos, seleccionar adecuadamente las bibliotecas de lenguajes de programación y proponer soluciones funcionales a problemas contextualizados en la Ingeniería Civil. Asimismo, el desarrollo del proyecto evidenció diferencias en la forma en que los estudiantes abordaron la resolución de problemas al emplear C o Python, aportando información relevante para el análisis comparativo planteado en este capítulo.

Por otro lado, la práctica docente evidenció que la implementación del ABP favorece un ambiente de aprendizaje activo y reflexivo, en el que los estudiantes asumen un rol protagónico en su formación. La integración del análisis comparativo de lenguajes de programación, actividades contextualizadas y evaluaciones continuas contribuyó al fortalecimiento del pensamiento computacional, la autonomía y la aplicación de conocimientos, aspectos fundamentales en la formación inicial de los futuros ingenieros civiles.

2.1 Implementación del análisis comparativo

La implementación del análisis comparativo se realizó en la asignatura Programación Aplicada a la Ingeniería y fue llevada a cabo durante el ciclo escolar 2025-B, con una duración de 17 semanas, dirigida a estudiantes de primer semestre de la carrera de Ingeniería Civil del Centro Universitario de Ciencias Exactas e Ingenierías. Los 5 grupos sometidos a evaluación estuvieron conformados por aproximadamente 30 estudiantes cada uno, quienes participaron de manera activa en las actividades diseñadas bajo un enfoque ABP y con el uso de lenguajes C y Python.

La planeación del curso contempló una estructura progresiva que permitió introducir los contenidos de manera gradual, articulando los conceptos teóricos con actividades prácticas y tareas orientadas a la resolución de problemas propios del ámbito de la Ingeniería Civil. El desarrollo de la asignatura se organizó en 4 etapas, las cuales permitieron integrar los aprendizajes a lo largo del semestre y culminar con un proyecto integrador, en el que los estudiantes aplicaron de forma conjunta los conocimientos adquiridos.

El proceso de enseñanza-aprendizaje se estructuró en las siguientes etapas:

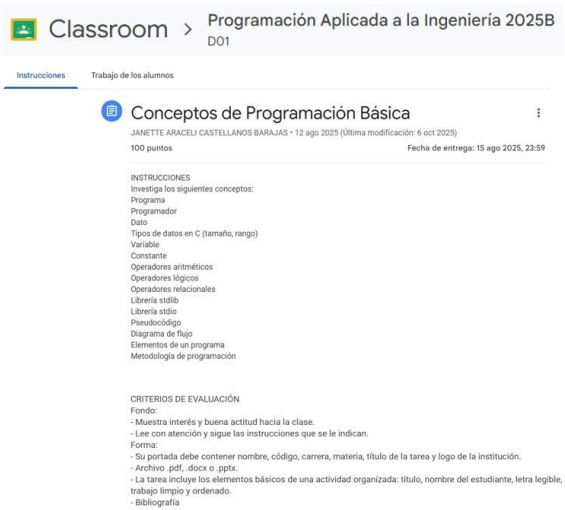
1. Planeación del curso: Se utilizó el programa de estudios que proporciona la institución educativa de la asignatura, que incluye objetivos de aprendizaje y contenidos temáticos. Asimismo, dicho programa cuenta con el total de horas, horas teóricas y prácticas que deben ser impartidas en un total de 17 semanas. Cada tema se orientó al desarrollo del razonamiento lógico y la comprensión de la programación aplicada a la Ingeniería Civil, considerando tanto el lenguaje C como Python, de acuerdo con la estrategia comparativa planteada para el ciclo escolar 2025-B. En la Figura 1 se puede observar el contenido temático sintético de la unidad de aprendizaje de Programación Aplicada a la Ingeniería, así como las horas de impartición del curso.

Figura 1
Contenido temático de la unidad de aprendizaje

Nombre de la Unidad de Aprendizaje: Programación Aplicada a la Ingeniería		
Total de horas: 60 horas	Teoría: 20 horas	Practica: 40 horas
Carrera: Ingeniería Civil		
Contenido Temático		
1. Conceptos básicos de programación estructurada		
2. Estructuras de control		
3. Arreglos		
4. Manejo de Funciones		
5. Estructuras de datos		

2. Diseño de actividades por tema: Durante el semestre, se diseñaron e impartieron clases con actividades teórico-prácticas en las que los estudiantes realizaron ejercicios guiados, prácticas individuales y actividades colaborativas. Estas actividades estuvieron diseñadas para reforzar conceptos como variables, estructuras de control, diseño de algoritmos y resolución de problemas, permitiendo a los estudiantes aplicar de manera inmediata los contenidos revisados en clase. En la Figura 2 se evidencia el ejemplo de la estructura de una de las actividades (instrucciones y criterios de evaluación).

Figura 2
Ejemplo de actividades (Classroom)



De manera complementaria, se diseñaron actividades prácticas por tema, las cuales fueron evaluadas considerando criterios como la correcta implementación del algoritmo, la claridad del código, el uso adecuado de estructuras de control y la capacidad para resolver problemas de forma lógica y estructurada. Estas actividades permitieron fortalecer el aprendizaje gradual y preparar a los estudiantes para el desarrollo del proyecto integrador. En la Figura 3 se muestra un algoritmo que permite leer archivos CSV con datos de un terreno, los cuales pueden ser utilizados para realizar cálculos y al mismo tiempo pueden graficarse desde Spyder y el uso de bibliotecas (funciones predefinidas) ya integradas; por el contrario, en la Figura 4 se muestra el código equivalente en C utilizando el compilador Codeblocks, el cual solo permite analizar los datos y realizar cálculos con ellos, sin tener la posibilidad de graficarlos de forma fácil y que se tendría que utilizar bibliotecas externas, dificultando su uso para principiantes.

Figura 3
Ejemplo de actividad realizada (Spyder)

```python
import numpy as np
import matplotlib.pyplot as plt

# 1. Leer datos del archivo CSV
data = np.loadtxt(
    r"C:\Users\jarac\OneDrive\Escritorio\Matlab Civiles\perfil_terreno.csv",
    delimiter=",",
    skiprows=1
)

x = data[:, 0]  # Distancia
y = data[:, 1]  # Elevación

# 2. Graficar perfil original del terreno
plt.figure()
plt.plot(x, y, 'b-o', linewidth=1.5, label="Terreno")
plt.xlabel("Distancia (m)")
plt.ylabel("Elevación (m)")
plt.title("Perfil Longitudinal del Terreno")
plt.grid(True)
plt.hold = True  # compatibilidad conceptual con MATLAB

# 3. Calcular pendientes entre puntos (%)
pendiente = np.diff(y) / np.diff(x) * 100

# 4. Determinar rasante (perfil proyectado)
# Pendiente máxima ±3%
pmax = 3 / 100
y_inicio = y[0]
y_rasante = y_inicio + pmax * x

# 5. Graficar rasante
plt.plot(x, y_rasante, 'r--', linewidth=2, label="Rasante")
plt.legend()

# 6. Calcular volúmenes de corte y relleno
delta_y = y_rasante - y
ancho = 10  # ancho de carretera (m)
```

```python
vol_corte = 0.0
vol_relleno = 0.0

for i in range(len(delta_y) - 1):
    base = x[i + 1] - x[i]
    h1 = delta_y[i]
    h2 = delta_y[i + 1]
    area_trap = base * (abs(h1) + abs(h2)) / 2 * ancho

    if np.mean([h1, h2]) > 0:
        vol_relleno += area_trap
    else:
        vol_corte += area_trap

# 7. Representar zonas de corte y relleno
for i in range(len(x) - 1):
    if delta_y[i] > 0:
        plt.fill(
            [x[i], x[i], x[i + 1], x[i + 1]],
            [y[i], y_rasante[i], y_rasante[i + 1], y[i + 1]],
            color='green', alpha=0.3
        )
    elif delta_y[i] < 0:
        plt.fill(
            [x[i], x[i], x[i + 1], x[i + 1]],
            [y[i], y_rasante[i], y_rasante[i + 1], y[i + 1]],
            color='red', alpha=0.3
        )

# Mostrar resultados
plt.show()

print(f"Volumen de corte: {vol_corte:.2f} m³")
print(f"Volumen de relleno: {vol_relleno:.2f} m³")
```

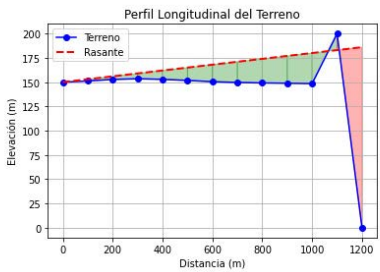


Figura 4
Ejemplo de actividad realizada (Codeblocks)

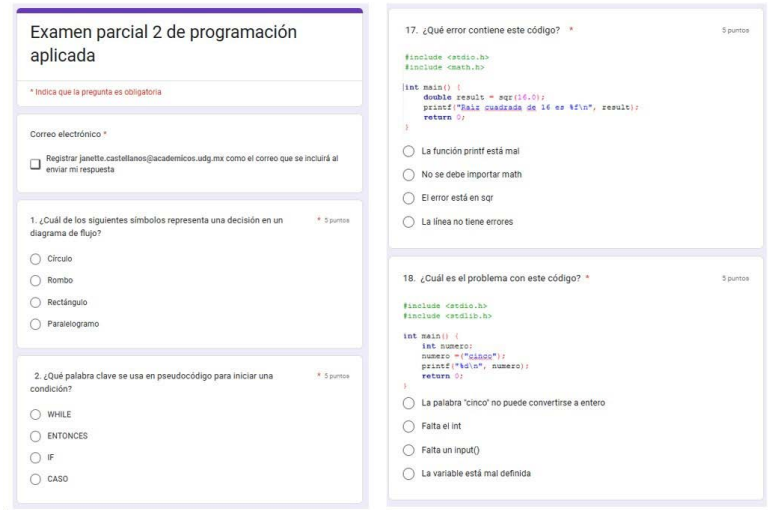
```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_PUNTOS 1000
#define PEND_MAX 0.03      // 3%
#define ANCHO 10.0         // ancho de carretera (m)

int main() {
    FILE *archivo;
    char linea[1024];
    double x[MAX_PUNTOS], y[MAX_PUNTOS];
    double y_rasante[MAX_PUNTOS];
    double delta_y[MAX_PUNTOS];
    int n = 0;

    // 1. Abrir archivo CSV
    archivo = fopen("C:/Users/jarac/OneDrive/Escritorio/perfil_terreno.csv", "r");
    if (archivo == NULL) {
        printf("Error al abrir el archivo.\n");
        return 1;
    }

    // 2. Leer y descartar encabezado
    fgets(linea, sizeof(linea), archivo);

    // 3. Leer datos
    while (fscanf(archivo, "%lf,%lf", &x[n], &y[n]) == 2) {
        n++;
        if (n >= MAX_PUNTOS) break;
    }
    fclose(archivo);
```

```c
    // 4. Calcular rasante
    double y_inicio = y[0];
    for (int i = 0; i < n; i++) {
        y_rasante[i] = y_inicio + PEND_MAX * x[i];
        delta_y[i] = y_rasante[i] - y[i];
    }

    // 5. Calcular volúmenes de corte y relleno
    double vol_corte = 0.0;
    double vol_relleno = 0.0;

    for (int i = 0; i < n - 1; i++) {
        double base = x[i + 1] - x[i];
        double h1 = delta_y[i];
        double h2 = delta_y[i + 1];
        double area_trap = base * (fabs(h1) + fabs(h2)) / 2.0 * ANCHO;

        if ((h1 + h2) / 2.0 > 0)
            vol_relleno += area_trap;
        else
            vol_corte += area_trap;
    }

    // 6. Mostrar resultados
    printf("Numero de puntos leidos: %d\n", n);
    printf("Volumen de corte: %.2f m3\n", vol_corte);
    printf("Volumen de relleno: %.2f m3\n", vol_relleno);

    return 0;
}
```

```
C:\Users\jarac\OneDrive\Escri
Numero de puntos leidos: 13
Volumen de corte: 0.00 m3
Volumen de relleno: 266000.00 m3

Process returned 0 (0x0)   execution time : 0.021 s
Press any key to continue.
```

3. Exámenes parciales (formularios digitales): Uno de los puntos clave en el análisis comparativo de los lenguajes de programación utilizados en el proceso de enseñanza-aprendizaje es el uso de exámenes parciales (formularios digitales). Dichos formularios de evaluación se utilizaron como una herramienta para medir el conocimiento adquirido por los estudiantes en cada unidad temática. Se aplicó un total de 4 exámenes en el semestre; estos instrumentos incluyeron preguntas de opción múltiple, ejercicios de análisis de código y resolución de problemas, permitiendo evaluar la comprensión de la lógica computacional y el manejo básico de los lenguajes de programación utilizados durante el curso. En la Figura 5 se puede observar un ejemplo de un cuestionario aplicado a los alumnos de Programación Aplicada a la Ingeniería.

Figura 5
Ejemplo de examen parcial (Google forms)



4. Proyecto integrador: Como complemento para la evaluación del análisis comparativo, se planteó un proyecto integrador en el que los estudiantes debían desarrollar una solución informática que incorporara todos los temas vistos durante el semestre. Este proyecto permitió evaluar la capacidad de los alumnos para integrar conocimientos, estructurar soluciones lógicas y desarrollar algoritmos de manera coherente y funcional. A

continuación, son presentados dos ejemplos del proyecto que realizaron estudiantes de los grupos sometidos a evaluación.

Ejemplo del proyecto en lenguaje C

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define PEND_MAX 0.03     // 3 %
#define ANCHO 10.0        // ancho de carretera (m)
#define MAX_PUNTOS 1000   // máximo de puntos del perfil

int leer_datos_csv(const char *ruta, double x[], double y[]) {
   FILE *archivo = fopen(ruta, “r”);
   if (archivo == NULL) {
      printf(“Error al abrir el archivo.\n”);
      return 0;
   }

   char linea[100];
   int n = 0;
   /* Saltar encabezado */
   fgets(linea, sizeof(linea), archivo);
   /* Leer datos */
   while (fscanf(archivo, “%lf,%lf”, &x[n], &y[n]) == 2) {
      n++;
      if (n >= MAX_PUNTOS) break;
   }

   fclose(archivo);
   return n;
}

void calcular_rasante(double x[], double y[], double y_rasante[],
                      double delta_y[], int n) {
   double y_inicio = y[0];
```

```
    for (int i = 0; i < n; i++) {
        y_rasante[i] = y_inicio + PEND_MAX * x[i];
        delta_y[i] = y_rasante[i] - y[i];
    }
}

void calcular_volumenes(double x[], double delta_y[], int n,
                        double *vol_corte, double *vol_relleno) {
    *vol_corte = 0.0;
    *vol_relleno = 0.0;

    for (int i = 0; i < n - 1; i++) {
        double base = x[i + 1] - x[i];
        double h1 = delta_y[i];
        double h2 = delta_y[i + 1];
        double area = base * (fabs(h1) + fabs(h2)) / 2.0 * ANCHO;
        if ((h1 + h2) / 2.0 > 0)
            *vol_relleno += area;
        else
            *vol_corte += area;
    }
}

int main() {
    double x[MAX_PUNTOS];
    double y[MAX_PUNTOS];
    double y_rasante[MAX_PUNTOS];
    double delta_y[MAX_PUNTOS];
    double vol_corte, vol_relleno;
    int n;
    n = leer_datos_csv("perfil_terreno.csv", x, y);
    if (n == 0) return 1;
    calcular_rasante(x, y, y_rasante, delta_y, n);
    calcular_volumenes(x, delta_y, n, &vol_corte, &vol_relleno);
    printf("Número de puntos: %d\n", n);
```

```
    printf("Volumen de corte: %.2f m³\n", vol_corte);
    printf("Volumen de relleno: %.2f m³\n", vol_relleno);
    return 0;
}
```

Ejemplo del proyecto en lenguaje Python

```
import numpy as np
import matplotlib.pyplot as plt
PEND_MAX = 0.03   # 3 %
ANCHO = 10.0      # ancho de carretera (m)
# ------------------------------------------------
def leer_datos_csv(ruta):
    """Lee un archivo CSV con encabezado: Distancia (m), Elevacion (m)"""
    data = np.genfromtxt(ruta, delimiter=",", skip_header=1)
    x = data[:, 0]
    y = data[:, 1]
    return x, y
# ------------------------------------------------
def calcular_rasante(x, y):
    """Calcula la rasante y la diferencia vertical"""
    y_inicio = y[0]
    y_rasante = y_inicio + PEND_MAX * x
    delta_y = y_rasante - y
    return y_rasante, delta_y
# ------------------------------------------------
def calcular_volumenes(x, delta_y):
    """Calcula volúmenes de corte y relleno usando trapecios"""
    vol_corte = 0.0
    vol_relleno = 0.0
    for i in range(len(x) - 1):
        base = x[i + 1] - x[i]
        h1 = delta_y[i]
        h2 = delta_y[i + 1]
        area = base * (abs(h1) + abs(h2)) / 2 * ANCHO
```

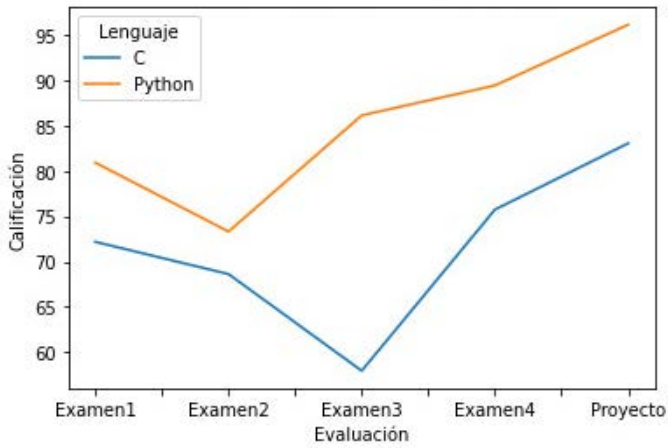
```
        if np.mean([h1, h2]) > 0:
            vol_relleno += area
        else:
            vol_corte += area
    return vol_corte, vol_relleno
# ------------------------------------------------
def graficar_perfil(x, y, y_rasante, delta_y):
    """Grafica perfil del terreno, rasante y zonas de corte y relleno"""
    plt.figure(figsize=(10, 5))
    plt.plot(x, y, 'b-o', label="Terreno", linewidth=1.5)
    plt.plot(x, y_rasante, 'r--', label="Rasante", linewidth=2)
    for i in range(len(x) - 1):
        if delta_y[i] > 0:  # relleno
            plt.fill([x[i], x[i], x[i + 1], x[i + 1]], [y[i], y_rasante[i], y_rasante[i
+ 1], y[i + 1]],
                color='green', alpha=0.3
            )
        elif delta_y[i] < 0:  # corte
            plt.fill([x[i], x[i], x[i + 1], x[i + 1]], [y[i], y_rasante[i], y_rasante[i
+ 1], y[i + 1]],
                color='red', alpha=0.3
            )
    plt.xlabel("Distancia (m)")
    plt.ylabel("Elevación (m)")
    plt.title("Perfil Longitudinal – Corte y Relleno")
    plt.grid(True)
    plt.legend()
    plt.show()
# ------------------------------------------------
def main():
    ruta = r"perfil_terreno.csv"
    x, y = leer_datos_csv(ruta)
    y_rasante, delta_y = calcular_rasante(x, y)
    vol_corte, vol_relleno = calcular_volumenes(x, delta_y)
    print(f"Número de puntos: {len(x)}")
```

```
    print(f"Volumen de corte: {vol_corte:.2f} m³")
    print(f"Volumen de relleno: {vol_relleno:.2f} m³")
    graficar_perfil(x, y, y_rasante, delta_y)
# ------------------------------------------------
if __name__ == "__main__":
    main()
```

3. Resultados

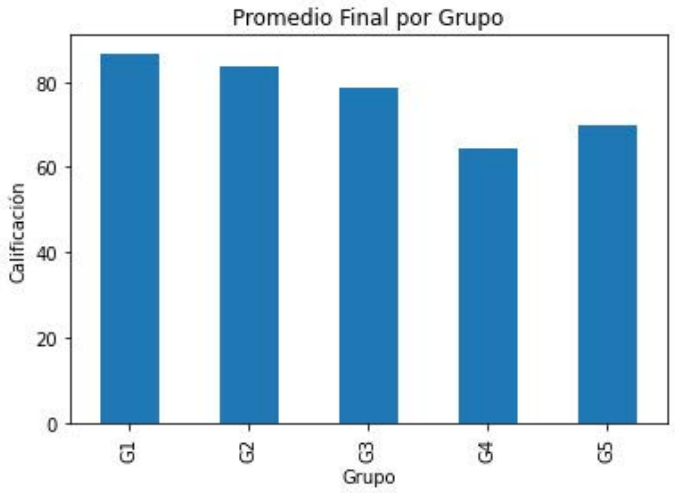
Los resultados obtenidos en el análisis comparativo entre los lenguajes C y Python en los procesos de enseñanza-aprendizaje de la asignatura de Programación Aplicada a la Ingeniería Civil demostraron un resultado satisfactorio a favor del lenguaje de programación Python. Dichos resultados fueron obtenidos del promedio de la evaluación aplicada mediante 4 exámenes parciales y un proyecto integrador; la muestra fue de 126 alumnos de una población de 146 alumnos. La cantidad de alumnos que fueron evaluados para el lenguaje C fue de 76 alumnos, mientras que para el lenguaje Python fue de 50 alumnos. En la gráfica de la Figura 6, se puede observar la tendencia del promedio por evaluación y lenguaje a tener un valor más elevado para el lenguaje de Python y con una característica notoria en el examen parcial 3, en la que los alumnos que utilizaron el lenguaje C obtuvieron un promedio de 60 comparado con un promedio de 87 que obtuvieron los alumnos del lenguaje Python. Esta diferencia de promedio se puede atribuir a que la sintaxis en lenguaje C para el uso de estructuras de control (if, for y while) es más compleja, ya que dicho examen estuvo enfocado principalmente en el tema anteriormente mencionado.

Figura 6
Promedio por evaluación y lenguaje.



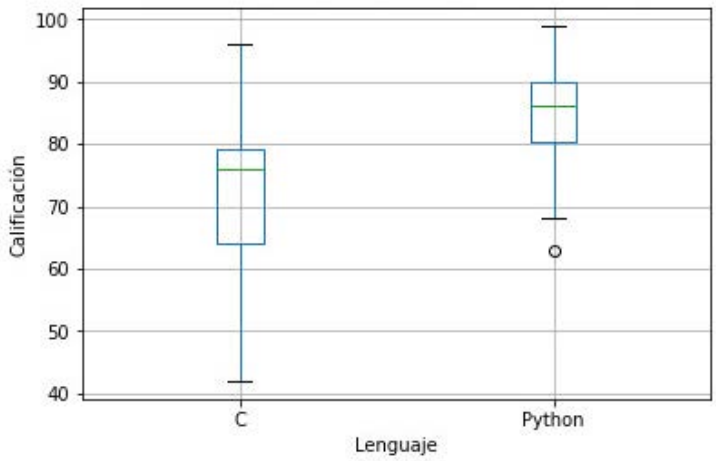
Adicionalmente, en la Figura 7, se puede observar el promedio final por grupo, en donde destacan los grupos G1 y G2 con mayor promedio que los grupos G3, G4 y G5. Esta diferencia de promedios entre los grupos nuevamente es favorecida por el uso del lenguaje Python en los grupos G1 y G2 en el proceso de enseñanza-aprendizaje de la asignatura de Programación Aplicada a la Ingeniería. Cabe aclarar que, aun siendo mayor la cantidad de alumnos pertenecientes a los grupos G3, G4 y G5 (26 alumnos más), no lograron superar el promedio de los otros dos grupos sometidos a evaluación y que incluso el grupo G4 fue el que obtuvo menor promedio.

Figura 7
Promedio por evaluación y lenguaje



La última gráfica presentada en la Figura 8 del análisis comparativo, que muestra la distribución del promedio final por lenguaje, representada mediante un diagrama de caja y bigotes, evidencia diferencias claras en el desempeño académico de los estudiantes que trabajaron con los lenguajes C y Python. En el caso de C, la mediana de las calificaciones se sitúa alrededor de valores intermedios y presenta una mayor dispersión, con calificaciones mínimas considerablemente más bajas, lo que indica una variabilidad significativa en el rendimiento estudiantil, posiblemente asociada a la complejidad sintáctica y conceptual del lenguaje para quienes se inician en la programación. Por el contrario, Python muestra una mediana más alta y una distribución más concentrada de las calificaciones, lo que refleja un desempeño general superior y más homogéneo, aun cuando se identifica un valor atípico de bajo rendimiento. En conjunto, estos resultados sugieren que Python facilita un aprendizaje inicial más accesible y consistente, mientras que C exige un mayor nivel de abstracción y precisión técnica, lo que puede impactar de manera más heterogénea en los resultados académicos.

Figura 8
Distribución del promedio final por lenguaje



La Tabla 1 muestra las métricas del promedio y la desviación estándar por grupo y lenguaje de programación, las cuales evidencian diferencias relevantes en el desempeño y la variabilidad de las calificaciones entre Python y C. Los grupos que trabajaron con Python (G1 y G2) presentan promedios más altos (86.680 y 83.640, respectivamente) y desviaciones estándar moderadas, lo que indica un rendimiento académico elevado y relativamente homogéneo entre los estudiantes. En contraste, los grupos que utilizaron el lenguaje C (G3, G4 y G5) muestran promedios inferiores, especialmente el grupo G4 con un promedio de 64.522, así como desviaciones estándar más elevadas, destacando nuevamente G4 con una desviación de 15.623, lo que refleja una mayor dispersión y heterogeneidad en los resultados. Aunque el grupo G3 presenta una desviación similar a los grupos de Python, su promedio sigue siendo menor, mientras que G5 se sitúa en un punto intermedio. En conjunto, estos datos sugieren que Python favorece un desempeño académico más alto y consistente en comparación con C, el cual presenta mayores dificultades de aprendizaje inicial que se reflejan en promedios más bajos y una mayor variabilidad en las calificaciones.

Tabla 1
Resultados de las métricas por grupo

Grupo	Lenguaje	Promedio	Desviación	Total Alumnos
G1	Python	86.680	7.256	25
G2	Python	83.640	9.013	25
G3	C	78.679	7.175	28
G4	C	64.522	15.623	23
G5	C	69.880	10.572	25

Por último, la Tabla 2 muestra las métricas de promedio y desviación estándar por lenguaje de programación; los resultados globales evidencian diferencias significativas en el desempeño académico de los estudiantes que trabajaron con C y Python. En el caso del lenguaje C, se observa un promedio general de 71.5 con una desviación estándar de 12.69, lo que indica no solo un rendimiento inferior, sino también una mayor dispersión en las calificaciones, reflejando diferencias marcadas entre los estudiantes. Por el contrario, Python presenta un promedio considerablemente más alto de 85.16 y una desviación estándar menor de 8.24, lo que sugiere un desempeño académico superior y más homogéneo. A pesar de que el grupo de C cuenta con un mayor número de alumnos, los resultados evidencian que Python favorece un aprendizaje más consistente y efectivo en contextos de enseñanza-aprendizaje en los principios de la programación, mientras que C tiende a generar mayores dificultades, manifestadas en calificaciones más bajas y una variabilidad más amplia en los resultados.

Tabla 2
Resultados de las métricas por lenguaje

Lenguaje	Promedio	Desviación	Total alumnos
C	71.500	12.686	76
Python	85.160	8.242	50

En resumen, los resultados del análisis comparativo entre los lenguajes de programación C y Python confirman que los estudiantes que trabajaron

con Python obtuvieron promedios más altos y una menor dispersión en sus calificaciones, lo que refleja un aprendizaje más consistente y homogéneo. En contraste, los grupos que utilizaron el lenguaje C presentaron promedios inferiores y desviaciones estándar más elevadas, evidenciando mayores dificultades y una variabilidad significativa en el desempeño académico. Estos resultados refuerzan la idea de que Python resulta más accesible y adecuado para la enseñanza inicial de la programación, aun siendo esta enseñanza a nivel ingeniería, mientras que C demanda un mayor nivel de abstracción y rigor técnico, lo cual impacta de manera diferenciada en los resultados de aprendizaje.

4. Conclusiones y recomendaciones

A partir del análisis integral de la información cuantitativa y gráfica obtenida del estudio realizado a los cinco grupos sometidos a evaluación, es posible establecer conclusiones sólidas sobre el impacto del lenguaje de programación utilizado en el desempeño académico de los estudiantes en contextos de enseñanza inicial de la programación. Los resultados muestran de manera consistente que los grupos que trabajaron con el lenguaje Python alcanzaron promedios de calificación significativamente más altos, acompañados de desviaciones estándar menores, lo que evidencia no solo un mejor rendimiento general, sino también una mayor homogeneidad en los aprendizajes logrados. Esta estabilidad en los resultados sugiere que Python facilita la comprensión de los conceptos fundamentales de programación, permitiendo que un mayor número de estudiantes alcance los objetivos de aprendizaje con menor dispersión en su desempeño, lo cual es atribuible a su sencillez en la sintaxis y el uso de bibliotecas enfocadas a la ingeniería.

Por el contrario, los datos correspondientes a los grupos que utilizaron el lenguaje C reflejan promedios más bajos y una variabilidad considerablemente mayor en las calificaciones, lo cual indica la presencia de dificultades diferenciadas entre los estudiantes. La amplitud de las desviaciones estándar y la dispersión observada en las distribuciones sugieren que, si bien algunos alumnos logran adaptarse adecuadamente al rigor del lenguaje, una proporción relevante enfrenta obstáculos asociados a su

sintaxis más estricta, al manejo explícito de memoria y a un mayor nivel de abstracción técnica. Estas características, aunque formativas desde una perspectiva ingenieril, pueden afectar negativamente la motivación y el rendimiento académico cuando se introducen en etapas tempranas del proceso formativo.

En conjunto, la evidencia analizada respalda la pertinencia pedagógica de Python como lenguaje introductorio, aunado a un enfoque ABP para la enseñanza de la programación en programas de ingeniería y áreas afines, al favorecer un aprendizaje más accesible, progresivo y consistente con una orientación práctica. No obstante, los resultados no desestiman el valor académico y formativo del lenguaje C, el cual resulta fundamental para el desarrollo de competencias profundas relacionadas con la programación de bajo nivel, la eficiencia y el control de recursos.

En este sentido, se recomienda una secuenciación curricular estratégica, en la que Python funcione como una herramienta inicial para consolidar el pensamiento computacional, la lógica algorítmica y la confianza del estudiante, permitiendo posteriormente la incorporación del lenguaje C en fases más avanzadas, cuando los alumnos cuenten con bases conceptuales sólidas. De esta manera, se promueve un proceso de enseñanza-aprendizaje más equilibrado, alineado con las necesidades cognitivas del estudiante y con los objetivos formativos de la educación en ingeniería.

Referencias

Asghar, J., Habib, S., Hussain, A., & Hussain, M. M. (2025). Comparación empírica de C++ y Python para la enseñanza de estudiantes introductorios de programación y su impacto en los resultados de aprendizaje. *Revista de Informática Computacional y Negocios*, *3*(1), 17–24. https://doi.org/10.65606/31/2025/39

Barrows, H. S. (1986). Una taxonomía de métodos de aprendizaje basados en problemas. *Educación Médica*, *20*(6), 481–486. https://doi.org/10.1111/j.1365-2923.1986.tb01386.x

Blank, W. E., Ed., & Harwell, S. Ed. (1997). *Prácticas prometedoras para conectar la escuela con el mundo real.*

Coll, C. (2006). *Psicología de la educación y práctica educativa: Necesidad y límites de la intervención educativa.* Graó.

Díaz Barriga Arceo, Frida. (2010). *Enseñanza situada: vínculo entre la escuela y la vida*. McGraw Hill.

Eastman, C. M. (2012). *BIM Handbook: una guía para modelar información de edificios para propietarios, directivos, diseñadores, ingenieros y contratistas.* Wiley.

Froese, T. M. (2010). El impacto de las tecnologías emergentes de la información en la gestión de proyectos de construcción. *Automatización en la construcción, 19*(5), 531–538. https://doi.org/10.1016/j.autcon.2009.11.004

Harwell, S. (1997). A*prendizaje basado en proyectos: prácticas prometedoras para conectar la escuela secundaria con el mundo real.* Universidad del Sur de Florida.

Kernighan, B. W., & Ritchie, D. M. (1988). *El lenguaje de programación C.* (2da ed.). Prentice Hall.

Martí Arias, J. (2010). *Educación y tecnologías.* Servicio de Publicaciones de la Universidad de Cádiz.

O'Brien, W. J. (2013). Gestión de la cadena de suministro en la construcción: Un marco de investigación. *Revista de Ingeniería y Gestión de la Construcción, 125*(5), 368–376.

Ritchie, D. M. (1993). *El desarrollo del lenguaje C. La Segunda Conferencia ACM SIGPLAN sobre la Historia de los Lenguajes de Programación*, 201–208. https://doi.org/10.1145/154766.155580

Robins, A., Rountree, J., & Rountree, N. (2003). Programación de aprendizaje y enseñanza: una revisión y discusión. *Educación en Informática, 13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Sebesta, R. W., Mukherjee, Soumen., & Bhattacharjee, A. Kumar. (2016). *Conceptos de lenguajes de programación.* Pearson Education Limited.

SEP. (2022). *Programa de Estudios de Educación Básica 2022.* Secretaría de Educación Pública. https://www.gob.mx/sep

UNESCO. (2019). *Marco de competencias de los docentes en materia de TIC*. Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura.

Van Rossum, Guido., & Drake, F. L. (2011). *El manual de referencia del lenguaje Python: para Python versión 3.2. Network Theory Ltd.*