# Actividad: Exploración de Lenguajes de Programación de Alto Nivel y Programación estructurada

## Identificación del trabajo

|  |  |
| --- | --- |
| 1. **Módulo:** | 1 |
| 1. **Asignatura:** | Fundamentos de Programación |
| 1. **Docente:** | Juan Simon Isidro |
| 1. **Estudiante:** | William Huera |
| 1. **Fecha de entrega**: | 03-09-2024 |

## Contexto

**Investigación Teórica**:

* Realiza una investigación sobre las características y ventajas de los lenguajes de programación de alto nivel, como Python, Java, y C#.
* Identifica los conceptos fundamentales de la programación estructurada, como la modularidad, la reutilización de código, y el uso de estructuras de control.
* Realiza una investigación sobre que es un IDE, su utilidad y proporciona ejemplos que soporten lenguajes de programación como Java, c#, Python, c++, Ruby, etc..

**Desarrollo de Programas**:

* Desarrolla dos programas sencillos en tres lenguajes de programación de alto nivel (por ejemplo, Python, Java y c#) que realicen una tarea específica.
* Implementa principios de programación estructurada en estos programas, utilizando funciones/métodos, bucles y estructuras condicionales.

**Documentación del Código**:

* Asegúrate de documentar el código adecuadamente, incluyendo comentarios que expliquen la funcionalidad de cada sección del programa.
* Realiza capturas de pantalla del proceso de desarrollo en el IDE o IDE's utilizados.

**Uso de IDEs**:

* Utiliza diferentes IDEs para escribir, compilar y ejecutar los programas (por ejemplo, PyCharm para Python, IntelliJ IDEA para Java Visual Studio Community).
* Documenta la experiencia de uso de cada IDE, resaltando las características que facilitan la programación estructurada (autocompletado, depuración, gestión de proyectos, etc.).

**Comparación y Análisis**:

* Compara las diferencias y similitudes entre los lenguajes de programación utilizados en términos de sintaxis, legibilidad, y facilidad de implementación de la programación estructurada.
* Analiza cómo cada IDE contribuyó a mejorar la eficiencia y la organización del desarrollo del programa.

## Problemas a resolver:

* Prepara una presentación que incluya:
* Una introducción a los lenguajes de programación de alto nivel y la programación estructurada.
* Descripción y demostración de los programas desarrollados.
* Comparación de las experiencias con los diferentes IDEs.
* Reflexiones sobre las ventajas y desafíos encontrados en el uso de lenguajes de alto nivel y herramientas de programación estructurada.

## Desarrollo:

**1. Características y Ventajas de los Lenguajes de Programación de Alto Nivel**

**Python**

* **Características:**
  + Sintaxis clara y sencilla.
  + Gestión automática de memoria.
  + Extensa librería estándar.
  + Dinámicamente tipado.
* **Ventajas:**
  + Alta productividad gracias a su legibilidad.
  + Amplia comunidad y recursos.
  + Compatible con múltiples paradigmas (procedural, orientado a objetos, funcional).
  + Versátil, utilizado en web, ciencia de datos, automatización, entre otros.

**Java**

* **Características:**
  + Sintaxis basada en C++.
  + Portabilidad gracias a la JVM (Java Virtual Machine).
  + Tipado estático.
  + Orientado a objetos.
* **Ventajas:**
  + Escrito una vez, ejecutado en cualquier lugar.
  + Robusto sistema de manejo de excepciones.
  + Extenso ecosistema de bibliotecas y frameworks.
  + Amplia adopción en aplicaciones empresariales.

**C#**

* **Características:**
  + Sintaxis similar a C++ y Java.
  + Integrado con la plataforma .NET.
  + Soporte robusto para programación orientada a objetos.
  + Tipado estático.
* **Ventajas:**
  + Fuerte integración con Windows.
  + Alto rendimiento en aplicaciones de escritorio y juegos.
  + Herramientas potentes como Visual Studio.
  + Buen soporte para el desarrollo de aplicaciones web con ASP.NET.

**2. Conceptos Fundamentales de la Programación Estructurada**

**Modularidad**

* Se refiere a dividir un programa en módulos o partes más pequeñas y manejables. Cada módulo cumple una función específica y puede ser desarrollado, probado y mantenido de forma independiente.

**Reutilización de Código**

* Implica usar código existente para nuevos propósitos, lo que ahorra tiempo y esfuerzo. Los módulos bien diseñados pueden ser reutilizados en diferentes partes de un programa o incluso en diferentes proyectos.

**Estructuras de Control**

* Son bloques de código que controlan el flujo de ejecución del programa. Incluyen:
  + **Condicionales:** Permiten tomar decisiones (if, else, switch).
  + **Bucles:** Permiten repetir una secuencia de instrucciones (for, while, do-while).

**3. Qué es un IDE y Ejemplos**

**IDE (Entorno de Desarrollo Integrado)**

Un IDE es un software que proporciona herramientas completas para escribir, probar, y depurar programas. Un buen IDE incluye:

* **Editor de código:** Para escribir el código fuente.
* **Depurador:** Para encontrar y corregir errores.
* **Compilador/Intérprete:** Para convertir el código en un programa ejecutable.
* **Gestor de proyectos:** Para organizar archivos y configuraciones del proyecto.

**Ejemplos de IDEs**

* **IntelliJ IDEA:** Soporta Java, Kotlin, Groovy, y Scala.
* **Visual Studio:** Soporta C#, VB.NET, C++, Python, y más.
* **PyCharm:** Especializado en Python, con soporte para Django.
* **Eclipse:** Multi-lenguaje, extensible para Java, C++, Python, etc.
* **RubyMine:** Especializado en Ruby y Rails.

**Desarrollo de Programas**

**Ejemplo de Programas en Python, Java y C#**

Tarea: Crear un programa que pida al usuario un número y determine si es par o impar.

|  |  |
| --- | --- |
| **Python:**  Programa para obtener numero par o impar en base al input del usuario | Terminal: |
| **Java:**  Programa en java, utilizando spring boot como framework y maven como compilador, para obtener numero par o impar en base al input del usuario | Terminal: |
| **C#:**  Programa para obtener numero par o impar en base al input del usuario | Terminal: |

Normativa, para la documentación del código

* En Python: # Esta función determina si un número es par o impar
* En Java: // Método para verificar si un número es par o impar
* En C#: // Verifica si el número es par o impar

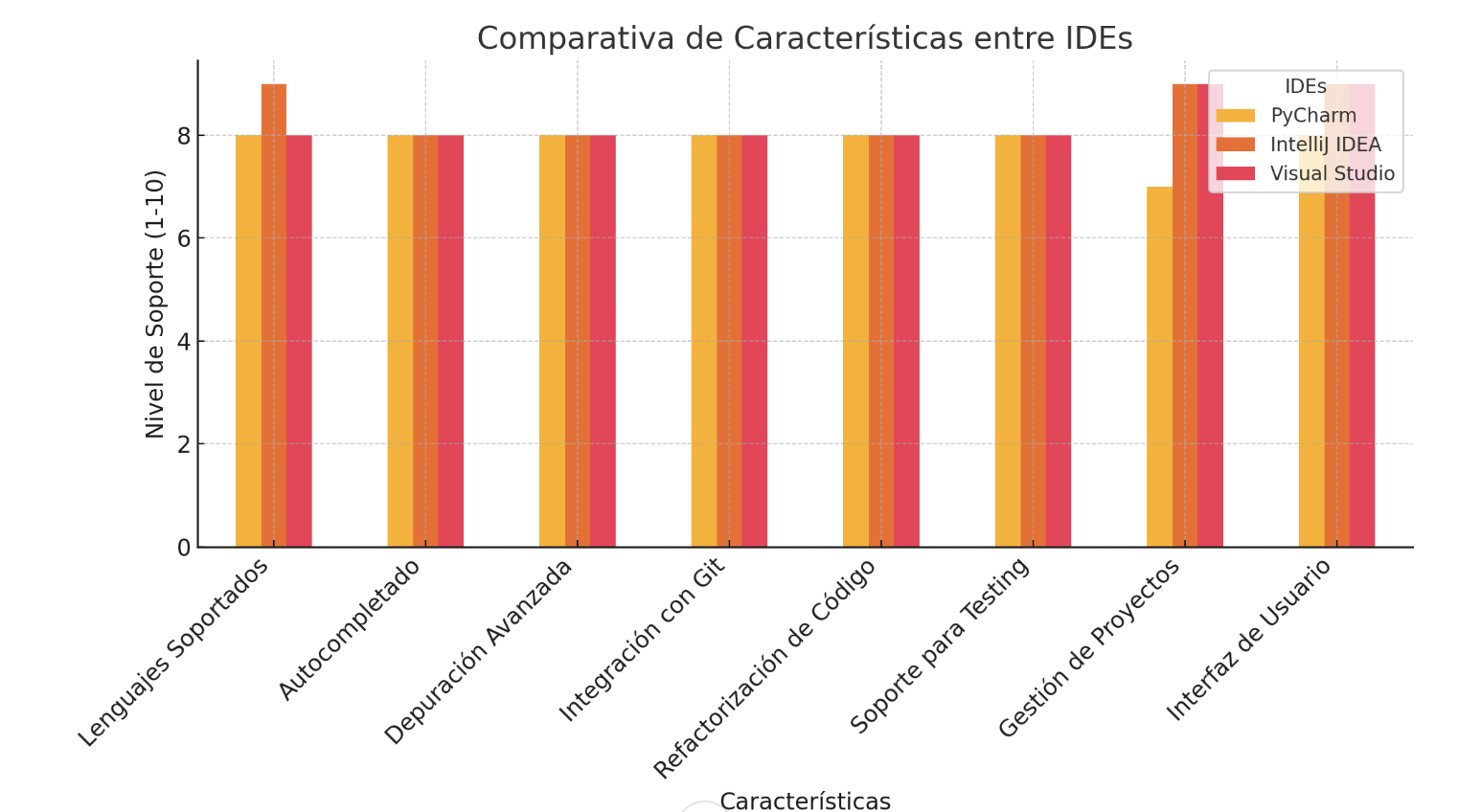
**Uso de IDEs**

* **PyCharm (JetBrains) para Python:**
  + Características como autocompletado, depuración avanzada, e integración con Git facilitan la gestión de proyectos complejos.
* **IntelliJ IDEA (JetBrains) para Java:**
  + Ofrece herramientas de refactorización, integración con Maven y Gradle, y soporte para pruebas unitarias con JUnit.
* **Visual Code para C#:**
  + Excelente para desarrollo en .NET, con características como IntelliSense, integración con Azure, y herramientas de depuración avanzadas.

**Comparación y Análisis**

* **Sintaxis y Legibilidad:**
  + **Python:** Es el más simple y legible, ideal para todo tipo de proyecto.
  + **Java y C#:** Sintaxis más verbosa, pero estructurada.
* **Facilidad de Implementación:**
  + **Python:** Menos código para lograr la misma funcionalidad.
  + **Java y C#:** Requieren más código, uso de frameworks (spring boot), pero ofrecen mayor control.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Características | PyCharm | IntelliJ IDEA | Visual Code |  |  |
| Lenguajes Soportados | Python, JavaScript, HTML/CSS, SQL | Java, Kotlin, Groovy, Scala, Python | C#, C++, VB.NET, Python, JavaScript | | |
| Autocompletado | Sí | Sí | Sí |  |  |
| Depuración Avanzada | Sí | Sí | Sí |  |  |
| Integración con Git | Sí | Sí | Sí |  |  |
| Refactorización de Código | Sí | Sí | Sí |  |  |
| Soporte para Testing | Sí (PyTest, Unittest) | Sí (JUnit, TestNG) | Sí (NUnit, MSTest) | |  |
| Gestión de Proyectos | Moderada | Alta | Alta |  |  |
| Interfaz de Usuario | Intuitiva y simple | Intuitiva y personalizable | Completa y rica en funcionalidades | | |



|  |  |
| --- | --- |
| Código fuente:  github | <https://github.com/whuera/fp_comparativa_lenguajes> |
| Video:  Drive | <https://drive.google.com/file/d/1cE4Q5RfeBebHH7UCsFiraUhpKrVwG6zd/view?usp=drive_link> |