Práctica 1

1) **1951 - 1955: Lenguajes tipo assembly**
   Características nuevas: Programación en lenguaje de bajo nivel, representación simbólica de instrucciones en código de máquina.
   - Lenguaje representativo: Ensamblador.

   1956 - 1960: FORTRAN, ALGOL 58, ALGOL 60, LISP
   Características nuevas:
   - Primeros lenguajes de alto nivel (FORTRAN, ALGOL 58).
   - Introducción de la programación estructurada (ALGOL 60).
   - Primer lenguaje funcional (LISP).

   1961 - 1965: COBOL, ALGOL 60, SNOBOL, JOVIAL
   Características nuevas:
   - Introducción de la programación orientada a negocios (COBOL).
   - Uso extendido de estructuras de control en la programación estructurada (ALGOL 60).
   - Expresiones regulares en procesamiento de texto (SNOBOL).

   1966 - 1970: APL, FORTRAN 66, BASIC, PL/I, SIMULA 67, ALGOL-W
   Características nuevas:
   - Introducción de la programación interactiva (BASIC).
   - Desarrollo de la programación orientada a objetos (SIMULA 67).
   - Manejo de matrices y operaciones matemáticas avanzadas (APL).
   - Introducción de características de programación multipropósito (PL/I).

   1971 - 1975: Pascal, C, Scheme, Prolog
   Características nuevas:
   - Introducción de la programación estructurada con tipado fuerte (Pascal).
   - Desarrollo de la programación de sistemas (C).
   - Introducción de la programación lógica (Prolog).
   - Enfoque en programación funcional con evaluación diferida (Scheme).

   1976 - 1980: Smalltalk, Ada, FORTRAN 77, ML
   Características nuevas:
   - Popularización de la programación orientada a objetos (Smalltalk).
   - Mayor énfasis en modularidad y seguridad en programación (Ada).
   - Introducción de inferencia de tipos en lenguajes funcionales (ML).

   1981 - 1985: Smalltalk 80, Turbo Pascal, PostScript
   Características nuevas:
   - Mejoras en la programación orientada a objetos (Smalltalk 80).
   - Optimización y compilación rápida en lenguajes estructurados (Turbo Pascal).
   - Lenguajes para la descripción de páginas y gráficos (PostScript).

1986 - 1990: FORTRAN 90, C++, SML
Características nuevas:

- Programación orientada a objetos en un lenguaje de alto rendimiento (C++).
- Mejora en programación científica y paralela (FORTRAN 90).
- Mejoras en lenguajes funcionales con inferencia de tipos más potente (SML).

1991 - 1995: TCL, PERL, HTML
Características nuevas:

- Introducción de lenguajes de scripting para automatización y manipulación de texto (TCL, PERL).
- Lenguajes para el desarrollo web y marcado de contenido (HTML).

1996 - 2000: Java, JavaScript, XML
Características nuevas:

- Independencia de plataforma con máquinas virtuales (Java).
- Introducción de la programación web interactiva en el navegador (JavaScript).
- Estándares de datos estructurados para intercambio de información (XML).

2) **Historia de Python:** Python fue creado por **Guido van Rossum** a finales de los años 80 y lanzado en **1991**. Su diseño se inspiró en lenguajes como ABC y C, priorizando la legibilidad del código y la simplicidad. Con el tiempo, Python evolucionó con versiones importantes como **Python 2 (2000)**, que introdujo mejoras en la administración de memoria y Unicode, y **Python 3 (2008)**, que modernizó la sintaxis y mejoró la consistencia del lenguaje. Actualmente, Python es ampliamente usado en desarrollo web, análisis de datos, inteligencia artificial y más.

Historia de Ruby: Ruby fue creado por **Yukihiro "Matz" Matsumoto** en **1995** en Japón, con el objetivo de ser un lenguaje fácil de usar y altamente flexible. Su diseño toma influencias de Perl, Smalltalk y Lisp. En **2004**, Ruby ganó popularidad global con el framework **Ruby on Rails**, que revolucionó el desarrollo web con su enfoque en la productividad y la simplicidad. A lo largo de los años, Ruby ha mantenido su filosofía de ser un lenguaje amigable y expresivo.

3) Atributos:
Ortogonalidad: Un lenguaje ortogonal tiene un conjunto reducido de conceptos que pueden combinarse de manera coherente sin efectos inesperados.
Expresividad: Un lenguaje expresivo permite escribir código conciso y claro sin sacrificar funcionalidad.
Legibilidad: Un código fácil de leer y entender reduce errores y facilita el mantenimiento. La sintaxis clara y el uso de convenciones ayudan a mejorar la legibilidad.
Simplicidad: Un lenguaje debe evitar características innecesarias y mantener una curva de aprendizaje accesible.
Eficiencia: Un buen lenguaje debe permitir la ejecución rápida del código y el uso eficiente de los recursos del sistema.

Portabilidad: Un lenguaje debe ser compatible con múltiples plataformas sin requerir cambios significativos en el código.
Modularidad: Un lenguaje debe permitir dividir el código en partes reutilizables, facilitando el desarrollo de proyectos grandes.
En general, **Python** es un lenguaje que cumple con muchas de estas características, destacándose por su legibilidad, expresividad y simplicidad.

4) Tipos de expresiones que permite python:
   - **Expresiones aritméticas**: Involucran operadores matemáticos (+, -, *, /, %, //, **). Ej: x = (5 + 3) * 2.
   - **Expresiones lógicas**: Utilizan operadores booleanos (and, or, not) para evaluar condiciones. Ej: es_mayor = (edad > 18 and tiene_permiso).
   - **Expresiones de comparación**: Comparan valores utilizando operadores como `<, >, ==, !=, <=, >=`. Ej: resultado = (x == y).
   - **Expresiones de asignación**: Asignan valores a variables. Ej: nombre = "Python".
   - **Expresiones condicionales (Ternarias)**: Devuelven un valor basado en una condición. Ej: mensaje = "Mayor de edad" if edad >= 18 else "Menor de edad".
   - **Expresiones lambda**: Son funciones anónimas que pueden usarse en una sola línea. Ej: doble = lambda x: x * 2.
   - **Expresiones de lista, diccionario y conjunto (comprehensions)**: Permiten construir estructuras de datos en una sola línea. Ej: cuadrados = [x**2 for x in range(10)].
   - **Expresiones de invocación de funciones**: Llaman a funciones y devuelven valores. Ej: resultado = suma(4, 5).

   Python proporciona varias herramientas para organizar el código de manera estructurada y modular:
   - Funciones.
   - Módulos.
   - Paquetes.
   - Clases y Objetos.
   - Espacios de nombres y alcance de variables.
   - Manejo de Excepciones.

Atributo	¿Python lo posee?	Justificación
Ortogonalidad	✅ Sí	Python tiene reglas de sintaxis simples y pocos casos especiales, lo que facilita la combinación de elementos del lenguaje de manera coherente.
Expresividad	✅ Sí	Permite escribir código claro y conciso, con estructuras como comprensiones de listas y expresiones lambda.
Legibilidad	✅ Sí	Su sintaxis clara y el uso obligatorio de indentación mejoran la legibilidad del código.
Simplicidad	✅ Sí	Tiene una sintaxis sencilla y fácil de aprender, con menos palabras clave y convenciones intuitivas.
Eficiencia	❌ No completamente	Aunque Python es eficiente en el desarrollo, su interpretación lo hace más lento que lenguajes compilados como C o Java.
Portabilidad	✅ Sí	Es multiplataforma y se ejecuta en distintos sistemas operativos sin cambios en el código.
Modularidad	✅ Sí	Ofrece soporte para módulos y paquetes, permitiendo estructurar el código en componentes reutilizables.

5) Ada es un lenguaje fuertemente tipado, lo que ayuda a prevenir errores en tiempo de compilación. Sus características incluyen:

Tipos de datos:

- **Tipos numéricos:** Integer, Float, Fixed-point y Modular (para aritmética modular).
- **Tipos de caracteres y cadenas:** Character y String.
- **Tipos enumerados:** Permiten definir conjuntos de valores simbólicos.
- **Tipos de acceso (punteros):** Son seguros y evitan errores de acceso a memoria.
- **Tipos derivados:** Permiten definir nuevos tipos basados en otros ya existentes.
- **Tipos de rango:** Se pueden restringir valores dentro de un rango.

Tipos de datos abstractos:

- Ada proporciona **paquetes (packages)** para encapsular datos y funciones, permitiendo modularidad y reutilización.

Estructuras de datos:

- **Registros (Records)**: Similares a las estructuras en C, permiten agrupar datos relacionados.
- **Arrays**: Se pueden indexar con cualquier tipo enumerado o numérico.
- **Listas enlazadas y estructuras dinámicas**: Implementadas usando punteros seguros.

Manejo de excepciones:

- Ada tiene un robusto sistema de manejo de excepciones, lo que permite capturar y manejar errores en tiempo de ejecución de manera estructurada.
- Se usan bloques **begin ... exception ... end** para manejar excepciones.
- Existen excepciones predefinidas como **Constraint_Error**, **Program_Error** y **Storage_Error**.

- Se pueden definir excepciones personalizadas.

Manejo de concurrencia:

- Ada tiene un modelo de concurrencia basado en **tareas (tasks)**, lo que facilita la programación concurrente y paralela sin necesidad de hilos explícitos.
- Se pueden definir tareas independientes que ejecutan procesos en paralelo.
- Se usa **rendezvous** para la sincronización entre tareas.
- Ada soporta mecanismos como **protected objects**, **semáforos** y **monitores**.

6) Java fue creado por **James Gosling** y su equipo en **Sun Microsystems** en **1995**. Su propósito inicial era desarrollar software para dispositivos electrónicos y sistemas embebidos. Sin embargo, debido a su portabilidad y robustez, Java se convirtió en un lenguaje ampliamente utilizado para aplicaciones empresariales, desarrollo web y móviles.
   Java revolucionó la web con varias innovaciones:
   - **Applets:** Pequeños programas embebidos en páginas web, que permitían contenido interactivo. Aunque hoy están obsoletos, fueron fundamentales en la evolución del contenido dinámico en la web.
   - **Java Servlets y JSP (JavaServer Pages):** Permitieron la creación de aplicaciones web dinámicas en el lado del servidor, mejorando el rendimiento respecto a CGI y otros métodos anteriores.
   - **Java EE (Enterprise Edition):** Introdujo tecnologías como EJB, JPA y JSF para desarrollar aplicaciones web escalables y seguras.
   - **Frameworks modernos:** Java impulsó frameworks como Spring y Hibernate, que facilitaron el desarrollo de aplicaciones web y empresariales.
   - **Android:** Aunque no es parte de la web, Java es el lenguaje base para el desarrollo de aplicaciones móviles en Android, expandiendo su impacto.
     Java es un lenguaje independiente de la plataforma.

   El código Java se compila en **bytecode** (un formato intermedio). Este **bytecode** es ejecutado por la **Java Virtual Machine (JVM)**, que se adapta a cada sistema operativo. Como resultado, el mismo código Java puede ejecutarse en Windows, Linux, macOS, y otros sistemas sin cambios.

7) Java está basado e influenciado por varios lenguajes de programación anteriores, entre los que destacan:
   - **C**: Java toma su sintaxis básica de C, lo que facilita la transición de programadores de C a Java. Sin embargo, elimina características problemáticas como los punteros explícitos y la manipulación directa de memoria.
   - **C++**: Java hereda la orientación a objetos de C++, pero simplifica el modelo eliminando herencia múltiple y mejorando la gestión de memoria con **recolección de basura (garbage collection)**.
   - **Smalltalk**: Influenció el modelo de orientación a objetos de Java, promoviendo la encapsulación, la modularidad y la independencia de plataforma.

- **Objective-C**: Aporta ideas sobre el envío dinámico de mensajes y la flexibilidad en la invocación de métodos.
- **Ada**: Inspiró algunas características de seguridad y robustez de Java, como la verificación estricta de tipos y el manejo de excepciones.

8) Los **applets** son programas escritos en **Java** que se ejecutaban dentro de un navegador web. Fueron populares en la década de los 90 y principios de los 2000 para agregar contenido interactivo a las páginas web.
<u>características:</u>
   - Se ejecutaban dentro de un navegador web que soportara **Java**.
   - No podían acceder directamente al sistema del usuario por razones de seguridad.
   - Se descargaban desde un servidor y se ejecutaban en la **Java Virtual Machine (JVM)** del navegador.
   - Eran utilizados para gráficos interactivos, juegos, visualización de datos, etc.

   Los applets quedaron obsoletos debido a problemas de seguridad y compatibilidad. La mayoría de los navegadores modernos ya no los soportan (Chrome, Firefox y Edge han eliminado el soporte para Java en la web).

   Los **servlets** son programas Java que se ejecutan en el **servidor** y generan contenido dinámico en una aplicación web. A diferencia de los applets, que se ejecutaban en el navegador, los servlets procesan solicitudes HTTP y generan respuestas, generalmente en formato HTML.
   <u>características:</u>
   - Se ejecutan en el servidor dentro de un contenedor web (como Apache Tomcat).
   - Manejan peticiones y respuestas HTTP, por lo que son la base de aplicaciones web en Java.
   - Son más seguros y eficientes que los applets, ya que no dependen del navegador del usuario.
   - Reemplazaron tecnologías más antiguas como CGI (Common Gateway Interface).

9) Un programa en **C** sigue una estructura general que incluye directivas de preprocesador, declaraciones de variables, funciones y el punto de entrada **main()**.
   Componentes principales de un programa en C:
   - **Directivas del preprocesador:** Instrucciones que se procesan antes de la compilación, como **#include** para importar bibliotecas.
   - **Declaración de macros y constantes:** Se pueden definir constantes con **#define** o **const**.
   - **Prototipos de funciones:** Declaraciones previas de funciones para modularidad.
   - **Funciones:** Implementaciones de las funciones del programa.
   - **Función main():** Es el punto de entrada del programa.

   En **C** no se pueden anidar funciones dentro de otras funciones. En C, todas las funciones deben definirse en el nivel superior del código fuente, fuera de cualquier otra función. No se permite definir una función dentro de otra.

10) En C, una **expresión** es una combinación de operandos (variables, constantes, funciones) y operadores que producen un resultado. Las expresiones pueden clasificarse en varios tipos según su función y estructura.
    - **Expresiones aritméticas:** Realizan operaciones matemáticas con operadores como +, -, *, /, %.
    - **Expresiones relacionales:** Comparan dos valores y devuelven un resultado booleano (1 para verdadero, 0 para falso).
    - **Expresiones lógicas:** Combinan valores booleanos usando operadores lógicos (&&, ||, !).
    - **Expresiones de asignación:** Utilizan el operador **=** para asignar valores a variables. También existen operadores de asignación compuesta (+=, -=, *=, /=, %=).
    - **Expresiones condicionales (operador ternario ? :):** Permiten evaluar una condición y devolver un valor según el resultado.
    - **Expresiones de incremento y decremento:** Modifican el valor de una variable en 1.
    - **Expresiones de bits:** Manipulan los bits individuales de un número usando operadores como &, |, ^, <<, >>.

    Cuando una expresión contiene múltiples operadores, C sigue **reglas de precedencia** para determinar el orden de evaluación.

11)

Python	Aplicaciones web, ciencia de datos, IA, automatización, scripting, aplicaciones de escritorio, servidores	Multi-paradigma (Orientado a Objetos, Imperativo, Funcional)	- Orientado a Objetos: Todo es un objeto. - Imperativo: Permite secuencias de instrucciones. - Funcional: Soporta funciones de orden superior, map, filter, lambda.
Ruby	Desarrollo web, automatización, aplicaciones empresariales, scripts de administración	Orientado a Objetos (también admite funcional e imperativo)	- Todo en Ruby es un objeto, incluso los números y funciones. - Soporta metaprogramación avanzada. - Permite bloques y closures, acercándolo a la programación funcional.
PHP	Aplicaciones web dinámicas, APIs, sistemas CMS (WordPress, Drupal)	Multi-paradigma (Principalmente Imperativo y Orientado a Objetos)	- Imperativo: Basado en secuencias de instrucciones. - Orientado a Objetos: Permite clases, herencia y polimorfismo. - Diseñado para la web: Integración nativa con HTML y servidores.

12) Python

- **Tipado de datos:** Dinámico. No se especifica el tipo de una variable cuando se declara.
- **Organización del programa:** Utiliza módulos y paquetes. Los programas se estructuran con archivos .py y se agrupan en carpetas con archivos __init__.py para definir paquetes.
- **Sintaxis:** Muy legible y concisa. Se utiliza indentación para definir bloques de código (sin llaves).
- **Manejo de errores:** Uso de excepciones con try, except, finally.
- **Comunidad y bibliotecas:** Gran ecosistema de bibliotecas para casi cualquier área, desde ciencia de datos hasta desarrollo web.
- **Orientado a objetos:** Todo en Python es un objeto.

Ruby

- **Tipado de datos:** Dinámico. No se requieren tipos explícitos para las variables.
- **Organización del programa:** Los programas se organizan en archivos .rb. Las clases y módulos se agrupan en archivos separados. Utiliza el concepto de gems (bibliotecas) para extender funcionalidad.
- **Sintaxis:** Elegante y expresiva. Fomenta el uso de "code blocks" y permite crear métodos de forma flexible.
- **Manejo de errores:** Uso de excepciones con begin, rescue, ensure.
- **Metaprogramación:** Permite la modificación del comportamiento del programa durante la ejecución (a través de métodos dinámicos).
- **Orientación a objetos:** Todo en Ruby es un objeto, incluso los tipos primitivos como números y cadenas.

PHP

- **Tipado de datos:** Dinámico. Los tipos de las variables no se declaran, aunque a partir de PHP 7, es posible declarar tipos de parámetros y valores de retorno en funciones.
- **Organización del programa:** Los programas en PHP se estructuran en archivos .php. Se puede organizar en funciones, clases y espacios de nombres (namespaces).
- **Sintaxis:** Similar a C y C++, con el uso de llaves {} para bloques de código.
- **Manejo de errores:** Usa excepciones con try, catch, finally.
- **Comunidad y bibliotecas:** Amplio soporte en el desarrollo web (especialmente para sistemas de gestión de contenido como WordPress).
- **Soporte de programación orientada a objetos:** PHP ofrece soporte completo para objetos y clases, con características como herencia, polimorfismo e interfaces.

Gobstone

- **Tipado de datos:** Dinámico. Gobstone permite trabajar con tipos de datos sin necesidad de especificar el tipo al declarar una variable.
- **Organización del programa:** Gobstone tiene una organización simple y está diseñado como un lenguaje para enseñar programación a estudiantes, con un enfoque en la manipulación de objetos gráficos llamados "gobstones".

- **Sintaxis:** Sencilla y amigable. Utiliza un estilo visual y basado en acciones en lugar de una sintaxis compleja.
- **Manejo de errores:** No tiene un sistema complejo de manejo de excepciones como otros lenguajes, pero permite controlar los errores básicos.
- **Paradigma:** Predominantemente imperativo.
- **Enseñanza:** Es un lenguaje educativo, diseñado para hacer más accesible la programación.

Processing

- **Tipado de datos:** Dinámico, similar a Java pero más enfocado a la programación creativa.
- **Organización del programa:** Los programas en Processing se organizan en sketches (bocetos), que son proyectos que incluyen el código fuente en archivos .pde.
- **Sintaxis:** Similar a Java, pero simplificada para facilitar el aprendizaje, especialmente en la creación de gráficos y visualizaciones.
- **Manejo de errores:** No tiene un sistema de manejo de errores tan estructurado, pero los errores se muestran de forma sencilla en el entorno de desarrollo.
- **Paradigma:** Principalmente imperativo y orientado a eventos (enfoque en la creación de gráficos interactivos).
- **Enfoque gráfico:** Su principal uso es en arte interactivo y visualizaciones, permitiendo crear gráficos de forma rápida y sencilla.
- **Comunidad:** Enfoque educativo y artístico, con una comunidad activa en áreas de diseño y visualización.

13) **JavaScript** es un lenguaje multi-paradigma. Esto significa que soporta varios estilos o enfoques de programación.
Los principales paradigmas que JavaScript soporta son:

- **Imperativo:** En este estilo, el programador define una secuencia de instrucciones que el computador debe ejecutar paso a paso. Es el paradigma más común y tradicional, y JavaScript lo permite fácilmente con estructuras como loops, condicionales, y asignaciones.
- **Orientado a objetos (OO)**: JavaScript permite trabajar con objetos y clases, aunque originalmente no tenía clases tradicionales. Desde ECMAScript 6 (ES6), JavaScript soporta clases, pero sigue siendo un lenguaje **basado en prototipos**, lo que significa que la herencia se da entre objetos directamente y no necesariamente a través de clases como en lenguajes como Java o C++.
- **Funcional:** JavaScript también soporta programación funcional, permitiendo el uso de funciones de primer orden, funciones anónimas, y métodos como map(), filter() y reduce(), lo que permite trabajar con funciones de manera más declarativa.
- **Event-driven (basado en eventos)**: JavaScript es muy usado para programación en entornos **basados en eventos**, especialmente en el navegador, donde las interacciones del usuario (clics, teclas, etc.) pueden desencadenar eventos que se manejan con funciones de callback.

Tipo de Lenguaje

- **Lenguaje de alto nivel:** JavaScript es un lenguaje de alto nivel porque se abstrae de detalles complejos del hardware, como la gestión de memoria y la administración de recursos del sistema, lo que facilita su programación.
- **Lenguaje interpretado:** JavaScript se ejecuta generalmente en tiempo real por el navegador o por un motor de ejecución (como Node.js), lo que significa que el código fuente no necesita ser compilado antes de ejecutarse.
- **Lenguaje de scripting:** Originalmente, JavaScript se desarrolló para agregar interactividad y funcionalidades dinámicas a las páginas web, por lo que se clasifica como un lenguaje de scripting. Su propósito inicial era manipular el DOM (Document Object Model) y hacer que las páginas web sean más interactivas y dinámicas.

14) Características:

- **Tipado dinámico**: JavaScript es un lenguaje **dinámicamente tipado**, lo que significa que no es necesario declarar el tipo de una variable. El tipo de datos se determina en tiempo de ejecución.
- **Conversión automática de tipos**: JavaScript realiza conversiones automáticas entre tipos en ciertas situaciones (coerción de tipos).
- **Declaración de variables:** Se utilizan las palabras clave var, let y const para declarar variables.
    - **var** tiene un alcance de función y es menos utilizado en la actualidad debido a su comportamiento confuso con el alcance (hoisting).
    - **let** tiene un alcance de bloque, por lo que es la mejor opción para la mayoría de los casos.
    - **const** también tiene un alcance de bloque y se usa para declarar variables cuyo valor no cambiará.
- **Manejo de excepciones:** JavaScript utiliza **try**, **catch** y **finally** para manejar errores de manera similar a otros lenguajes.
- **Funciones de primer orden:** En JavaScript, las funciones son ciudadanos de primer orden, lo que significa que pueden ser pasadas como parámetros, devueltas desde otras funciones y asignadas a variables.
- **Funciones anónimas**: JavaScript permite declarar funciones sin nombre, lo que es útil en el manejo de eventos o como argumentos en funciones.
- **Ámbito de bloque vs. ámbito de función:** El ámbito en JavaScript varía según si la variable fue declarada con var, let, o const.
    - **var:** Ámbito de función o global, lo que significa que no es adecuado para manejar bloques como estructuras de control o bucles.
    - **let y const:** Ámbito de bloque, limitado al bloque donde fueron declarados (por ejemplo, dentro de un bucle o una condición).
- **Asincronía**: JavaScript usa un modelo de ejecución **asíncrona** basado en eventos. Se utiliza para gestionar tareas como peticiones HTTP, temporizadores, etc.
- **Promesas:** Las promesas (**Promise**) permiten manejar operaciones asíncronas de manera más sencilla y controlada, evitando el **callback hell**.
- **Manipulación del DOM**: JavaScript es fundamental para interactuar con el **DOM** (Document Object Model) en las aplicaciones web, lo que permite crear dinámicamente o modificar contenido HTML y reaccionar a eventos.

- **Objetos**: JavaScript es un lenguaje orientado a objetos, pero con un enfoque en la **herencia prototípica** (en lugar de la herencia clásica basada en clases como en Java).
- **Clases (ES6)**: A partir de ECMAScript 6, JavaScript incluye **clases** para facilitar la creación de objetos y la herencia.
- **Destructuración**: JavaScript permite extraer valores de arrays u objetos de manera sencilla y directa.
- **Módulos**: Con **ES6**, JavaScript incluye un sistema de módulos nativo que permite dividir el código en archivos más pequeños y reutilizables, con la posibilidad de **importar** y **exportar** funciones, clases u otros elementos.