El Impacto de las Nuevas Tecnologías en la Sociedad: Visualización del Futuro.

Jonatahn F. Guachamín

Ingeniería en Inteligencia Artificial, Universidad Internacional del Ecuador

Lógica de Programación

MSc. Lilian M. Aman

Diciembre 21 de 2025

1. Introducción al Entorno de Desarrollo

En la última década, hemos sido testigos de un avance vertiginoso en el ámbito de las nuevas tecnologías. Desde la llegada de los smartphones hasta la proliferación de la inteligencia artificial, estas innovaciones han transformado radicalmente la forma en que vivimos, trabajamos y nos relacionamos. La tecnología no solo ha facilitado tareas cotidianas, sino que también ha abierto nuevas oportunidades y desafíos que requieren una adaptación constante por parte de la sociedad.

En este contexto, es fundamental entender cómo estas herramientas están moldeando nuestro entorno y qué implicaciones tienen para el futuro. Las nuevas tecnologías han permeado todos los aspectos de nuestra vida diaria. La conectividad instantánea, la automatización de procesos y el acceso a grandes volúmenes de información han cambiado nuestras expectativas y comportamientos (Martes Tecnológico, 2025).

1.1. Python

Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo (Amazon Web Services, 2025).

Los beneficios de Python incluyen los siguientes:

Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis básica similar a la del inglés.

Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con muchos otros lenguajes.

Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.

Los desarrolladores pueden utilizar Python fácilmente con otros lenguajes de programación conocidos, como Java, C y C++.

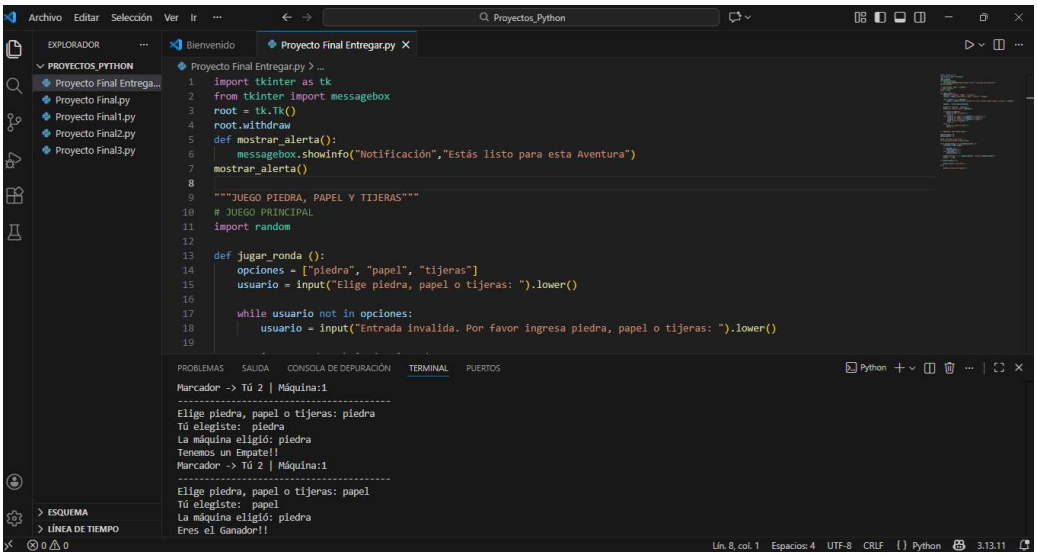
La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.

Hay muchos recursos útiles disponibles en Internet si desea aprender Python. Por ejemplo, puede encontrar con facilidad videos, tutoriales, documentación y guías para desarrolladores.

Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix.

Ilustración 1

Lógica de Programación Python

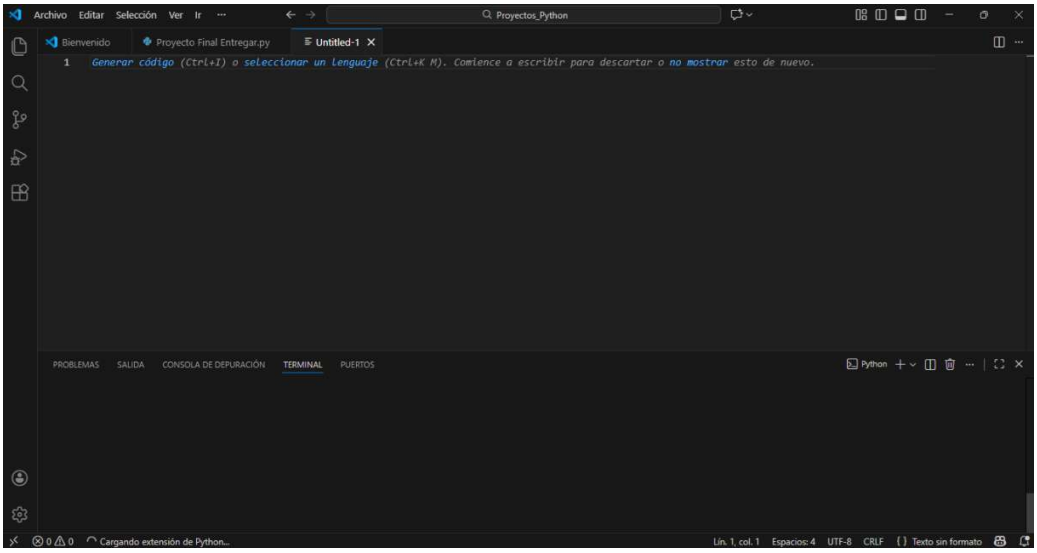


1.2. Editor de Código

Existen una gran cantidad de Editores de código, los cuales permiten utilizar correctamente la sintaxis y la lógica de los lenguajes de programación, el que se utilizó durante el desarrollo de la materia fue Visual Studio Code desarrollado por Microsoft para Windows, Linux, macOS y Web, debido a que apoya a una escritura de código más intuitiva, identifica palabras claves del lenguaje y genera el auto espaciado que requiere el lenguaje en los bloques de código.

Ilustración 2

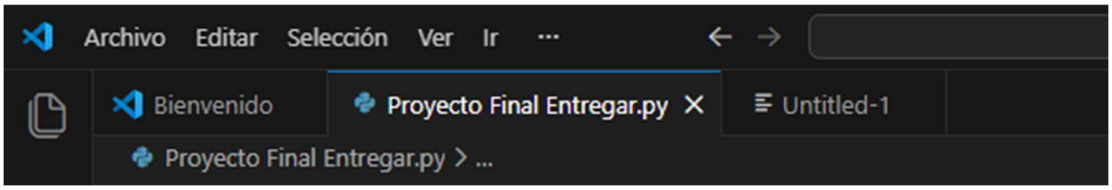
Pantalla de Visual Studio Code



Es importante tener en cuenta que el archivo debe tener la extensión.py eso lo identifica como un archivo de tipo Python.

Ilustración 3

Extensión.py



1.3. La depuración

Saber depurar nos va a permitir entender de una manera adecuada donde y debido a qué se producen los errores dentro de un programa. La gestión adecuada de errores te va a permitir ahorrar tiempo de desarrollo y mejorar tu sintaxis.

2. Manejo de Datos

Los tipos de datos en Python se dividen en varias categorías, cada una adaptada para manejar tipos específicos de datos y operaciones.

Comprender estas categorías es un ejercicio teórico y una habilidad práctica que nos ayudará a escribir código eficaz y sin errores. Seleccionar el tipo de datos correcto puede influir significativamente en el uso de la memoria, la velocidad de cálculo y la claridad del código.

Exploremos estas categorías en detalle, destacando sus casos de uso y aplicaciones.

Tipos de Datos:

En mi proyecto como tal utilizo tipo de datos enteros, booleanos y de cadenas de texto para ejecutar ciertas acciones pertinentes, no se requiere el uso de los otros tipos de Datos, pero cito un ejemplo de cómo se los podría utilizar.

int (Enteros): Números completos, positivos o negativos, sin parte fraccionaria.

Ejemplo: edad = 25, temperatura = -3.

Ilustración 4

Tipo de Datos

```
40   usuario_puntos = 0
41   maquina_puntos = 0
```

float (Flotantes): Números con punto decimal. Se usa el punto. como separador decimal.

Ejemplo: precio = 19.99, pi = 3.14159.

str (Cadenas de texto): Secuencia de caracteres (letras, números, símbolos) entre comillas simples (') o dobles (").

Ejemplo: nombre = "Ana", mensaje = '¡Hola Mundo!'.

Ilustración 5

Cadena de texto

```
while usuario not in opciones:
    usuario = input("Entrada invalida. Por favor ingresa piedra, papel o tijeras: ").lower()
```

```
print(f"Tú elegiste:  {usuario}")
print(f"La máquina eligió: {maquina}")
```

bool (Booleanos): Representa valores de verdad, solo puede ser True o False. Fundamental para la toma de decisiones.

Ejemplo: es mayor = (10 > 5) (sería True), caso contrario sería = False.

Ilustración 6

Uso de Booleanos

```
if usuario == maquina:
    print("Tenemos un Empate!!")
    return 0
elif (usuario == "piedra" and maquina == "tijeras") or \
     (usuario == "papel" and maquina == "piedra") or \
     (usuario == "tijeras" and maquina == "papel"):
     print ("Eres el Ganador!!")
```

Variables

Cada lenguaje de programación tiene su propia nomenclatura para llamar a las variables, algunos lenguajes de programación son sensibles a mayúsculas y minúsculas, en el caso de Python hay ciertas reglas que un programador debe seguir para nombrar variables.

Python está abierto a soportar cualquier tipo de convención de nombramiento de variable, pero algo que si es importante es que se debe ser coherente con la forma de nombrar sus variables.

El nombre de una variable debe:

Empezar con un carácter alfabético.

No contener más caracteres que letras, números y guiones bajos (_).

No contener espacios.

El nombre de la variable debe tener relación con su contenido.

Operadores

Utilice un operador de repetición que es el asterisco (*), que se usa para repetir secuencias como cadenas de texto un número específico de veces, creando una nueva secuencia con el contenido repetido

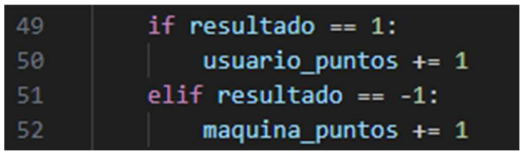
Ilustración 7

Uso de Operadores

También utilice operadores de asignación como el =, que sirve para asignar un valor a una variable, guardando datos para su uso posterior y a su vez emplee un operador compuesto como +=.

Ilustración 8

Uso de Operadores de asignación



3. Diagramas de Casos de Uso

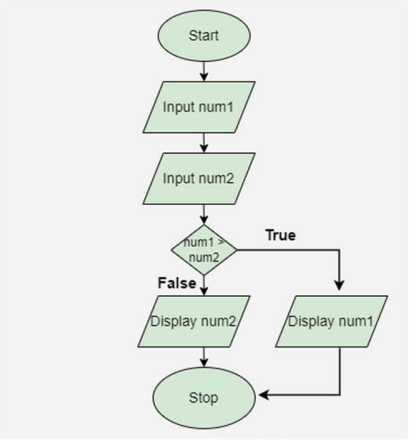
Un diagrama de casos de uso UML es la forma principal de definir los requisitos del sistema/software para un programa nuevo en desarrollo. Los casos de uso especifican el

comportamiento esperado (qué), pero no el método exacto para lograrlo (cómo). Una vez especificados, los casos de uso pueden representarse tanto textual como visualmente (es decir, mediante un diagrama de casos de uso). Un concepto clave del modelado de casos de uso es que nos ayuda a diseñar un sistema desde la perspectiva del usuario final. Es una técnica eficaz para comunicar el comportamiento del sistema en términos comprensibles para el usuario, especificando todo el comportamiento del sistema visible externamente (Visual Paradigm, 2025).

3.1. Ventajas de los diagramas de uso

- Facilita la comprensión de ideas y sistemas complejos
- Convierte el código complejo en un diagrama visual
- Mantiene a todo el mundo en la misma página
- Permite a los desarrolladores ver la imagen global de un sistema
- Ayuda a los no programadores a entender los procesos y funcionalidades del software
- Utilizar una notación común significa que cualquier programador puede entenderlo
- **Ilustración 9**

  Diagramas de Flujo

- 

4. Software a Desarrollar y Pasos de Resolución de Problemas

4.1. Desarrollar el Juego de Piedra, Papel o Tijera.

Identificar el Problema

Problema identificado: El comportamiento y los procesos mentales de los usuarios

Comprender el problema

Analizar la situación, detectando las causas:

- La tendencia a no repetir los movimientos
- Tendencia a favor o en contra de un elemento
- La idea de esperar o anticipar que una situación se repita o se mantenga igual, sin cambios.

Identificar Soluciones Alternativas

- Ser lo más aleatorio e impredecible posible.
- No prestes atención a lo que sucedió en la ronda anterior.
- Analiza los patrones de tu oponente

Seleccionar la mejor Solución

Ser lo más. aleatorio e impredecible posible.

Listar los pasos de la solución seleccionada

- Mantener un patrón corporal neutro y relajado.
- Tener las manos ocultas para nuestro oponente.
- Ejecutar movimientos aleatorios e impredecibles.

Evaluar y probar la solución del Problema

- Después de 3 rondas de juego en el que se aplicó este proceso:
- Logró cambiar el comportamiento a la hora de abordar el juego.

- Se desarrolló un ámbito de relajación al momento de ejecutar un movimiento.
- Se obtuvo más victorias al realizar movimientos analíticos y aleatorios.

Ilustración 10

Juego Piedra, Papel y Tijeras

5. Método de Resolución de Problemas mediante Programación

Problema identificado: El comportamiento y los procesos mentales de los usuarios.

5.1. Análisis del Problema (Entendimiento)

Definición Clara: El programa debe desarrollar el juego de Piedra, Papel y Tijeras de una manera sencilla, teniendo especificado un mensaje claro de las opciones y de las reglas que se ejecutarán.

Objetivos: crear un Programa atractivo y de fácil uso para los usuarios, que les permita ejecutarlos n veces sin miedo al aburrimiento.

Entradas y Salidas: tener presente el comportamiento de las diversas personalidades de los usuarios y hacerlo accesible y agradable para grandes y pequeños.

Restricciones (Alcance): Como etapa Inicial desarrollar un programa 100 por ciento ejecutable y con una configuración sencilla al alcance de todos. Las limitaciones que se puede tener en cualquier proyecto de programación es el factor económico y el tiempo para desarrollarlo.

5.2. Diseño del Algoritmo (La Receta)

Divide y Vencerás:

- El comportamiento y expectativas de grandes y pequeños.
- El uso de términos.
- Secuencia de ejecución del programa.
- Factor sorpresa (competitividad).

Pasos Lógicos:

- Implementar una programación de fácil ejecución, con términos sencillos y cómodos de ir desarrollando.
- Tener un orden claro y clave a la hora de ejecutar el programa.

Identifica Patrones:

- Dejar de lado lo tradicional e implementar un tipo de reto competitivo al implementar condiciones para que uno de los usuarios sea el vencedor.

5.3. Implementación (Codificación)

Codifica por Partes: Resolver primero las partes más sencillas como tener claro y mantener las reglas ya establecidas y que son de conocimiento de la mayor parte de los usuarios.

- Tener una introducción clara con las opciones específicas.
- El desarrollo del programa debe ser fluido e interesante.
- La parte final debe visualizar un resultado preciso.

Usa Recursos: implementar acciones y comandos diferentes.

5.4. Verificación y Depuración (Afinando)

Prueba y Error: se verificará durante la ejecución se buscará errores.

Replantear: Tener la mentalidad abierta y si algo no funciona o no está acorde al desarrollo del programa se debe tener una estrategia y buscar nuevas opciones.

5.5. Juego Piedra, Papel y Tijeras

Análisis:

Objetivo: ejecutar de forma fluida y sencilla el juego Piedra, Papel y Tijeras implementando funciones y comandos acordes a la lógica de programación del Juego.

Entrada: el usuario ingresar piedra, papel o tijeras (solo una opción).

Desarrollo: el juego se ejecutará de forma aleatoria hasta que el usuario o la maquina alcancen 3 victorias, ejecutándose un contador para ir visualizando cómo va el marcador.

Salida: se visualizará un mensaje final, indicando si ganó el usuario o la máquina.

Restricciones: solo se puede ingresar comandos alfabéticos y no numéricos.

Algoritmo (Python):

Ilustración 11

Lógica de programación Python

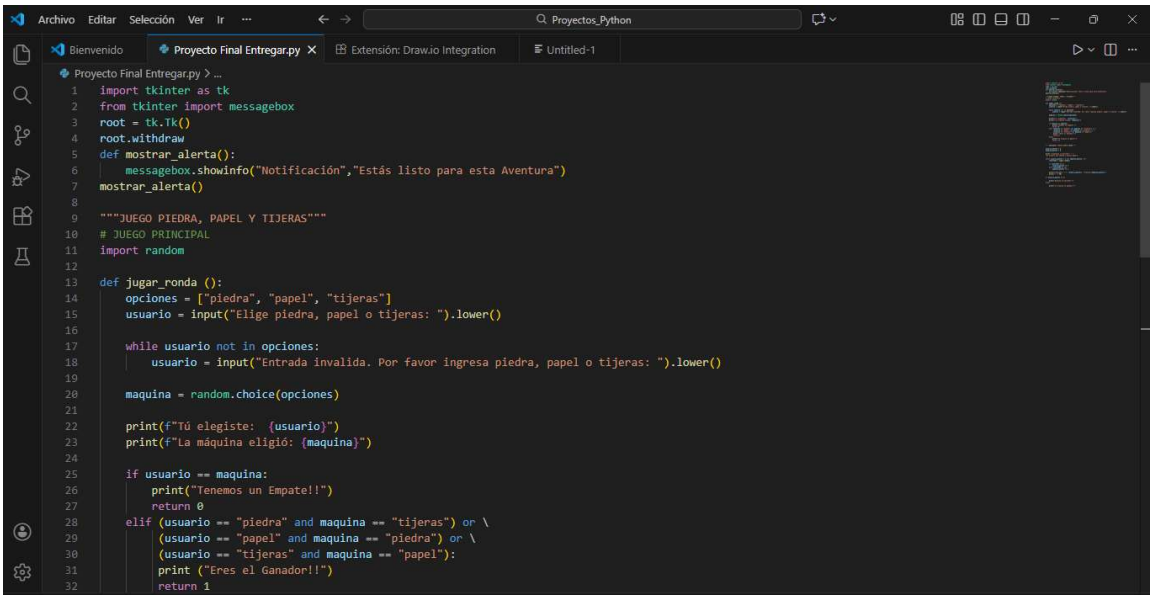
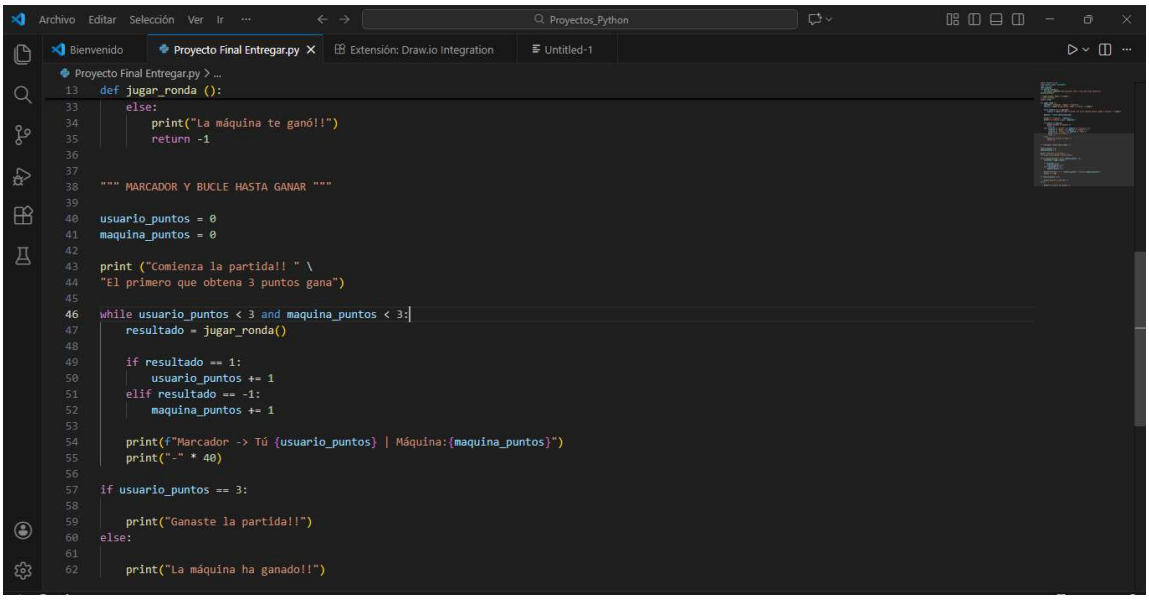


Ilustración 12

Lógica de Programación Python

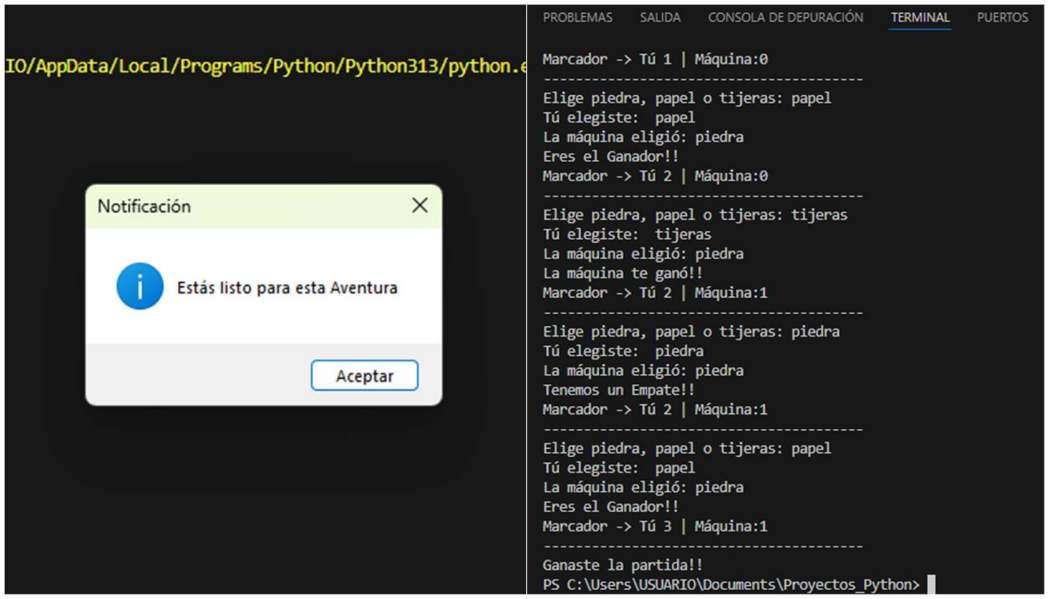


Codificación: Escribir el código en Python usando lo indicado en clases y otros métodos adicionales.

Depuración: Probar con diferentes valores, para ver si se desarrolla de forma correcta y ver que restricciones tiene como tal el programa (figuras, números, comandos alfanuméricos, etc.).

Ilustración 13

Ejecución del Juego Piedra, Papel o Tijeras



La clave es la planificación para entender el problema a fondo, el diseño del algoritmo, y la flexibilidad para ajustar el plan.

6. Condicionales

6.1. Operadores

Los operadores en Python son símbolos que realizan operaciones sobre valores (operandos) para manipular datos, comparar, tomar decisiones y construir lógica, incluyendo tipos como aritméticos (+, -, *), comparación (>, <, ==), lógicos (and, or, not), asignación (=, +=), esos son algunos de los operadores que emplee en el desarrollo del programa, teniendo en cuenta su uso en puntos específicos.

Ilustración 14

Uso de Operadores

```
while usuario_puntos < 3 and maquina_puntos < 3:
    resultado = jugar_ronda()
```

6.2. If (Expresión)

La expresión normalmente utiliza operadores lógicos y operadores relacionales para poder expresar la condición a ser evaluada.

Ilustración 15

Uso del If

```
if usuario == maquina:
    print("Tenemos un Empate!!")
```

```
if resultado == 1:
    usuario_puntos += 1
```

6.3. While

El uso del while nos permite ejecutar una sección de código repetidas veces, de ahí su nombre. El código se ejecutará mientras una condición determinada se cumpla.

Ilustración 16

Uso del While

```
while usuario not in opciones:
    usuario = input("Entrada invalida. Por favor ingresa piedra, papel o tijeras: ").lower()
```

6.4. For

El for es un tipo de bucle, parecido al While, pero con ciertas diferencias. La principal es que el número de iteraciones de un “for” está definido de antemano, mientras que en un while no. En mi proyecto no era necesario el implementar el uso de este bucle.

7. Estructura de Datos y Funciones

7.1. Tuplas

Las tuplas en Python o tuples son muy similares a las listas, pero con dos diferencias. Son inmutables, lo que significa que no pueden ser modificadas una vez declaradas, y en vez de inicializarse con corchetes se hace con (). Dependiendo de lo que queramos hacer, las tuplas pueden ser más rápidas (El Libro De Python, 2025).

7.2. Listas

Las listas en Python son uno de los tipos o estructuras de datos más versátiles del lenguaje, ya que permiten almacenar un conjunto arbitrario de datos. Es decir, podemos guardar en ellas prácticamente lo que sea (El Libro De Python, 2025).

7.3. Diccionarios

Un diccionario en Python es una colección de elementos, donde cada uno tiene una llave key y un valor value. Los diccionarios se pueden crear con paréntesis {} separando con una coma cada par key: value (El Libro De Python, 2025).

7.4. Funciones

Las funciones en Python son bloques de código reutilizables que realizan tareas específicas, ayudando a organizar, modularizar y simplificar programas largos, definiéndose con def, aceptando parámetros (entradas) y opcionalmente devolviendo valores con return, y se ejecutan al ser "llamadas" por su nombre.

Ilustración 17

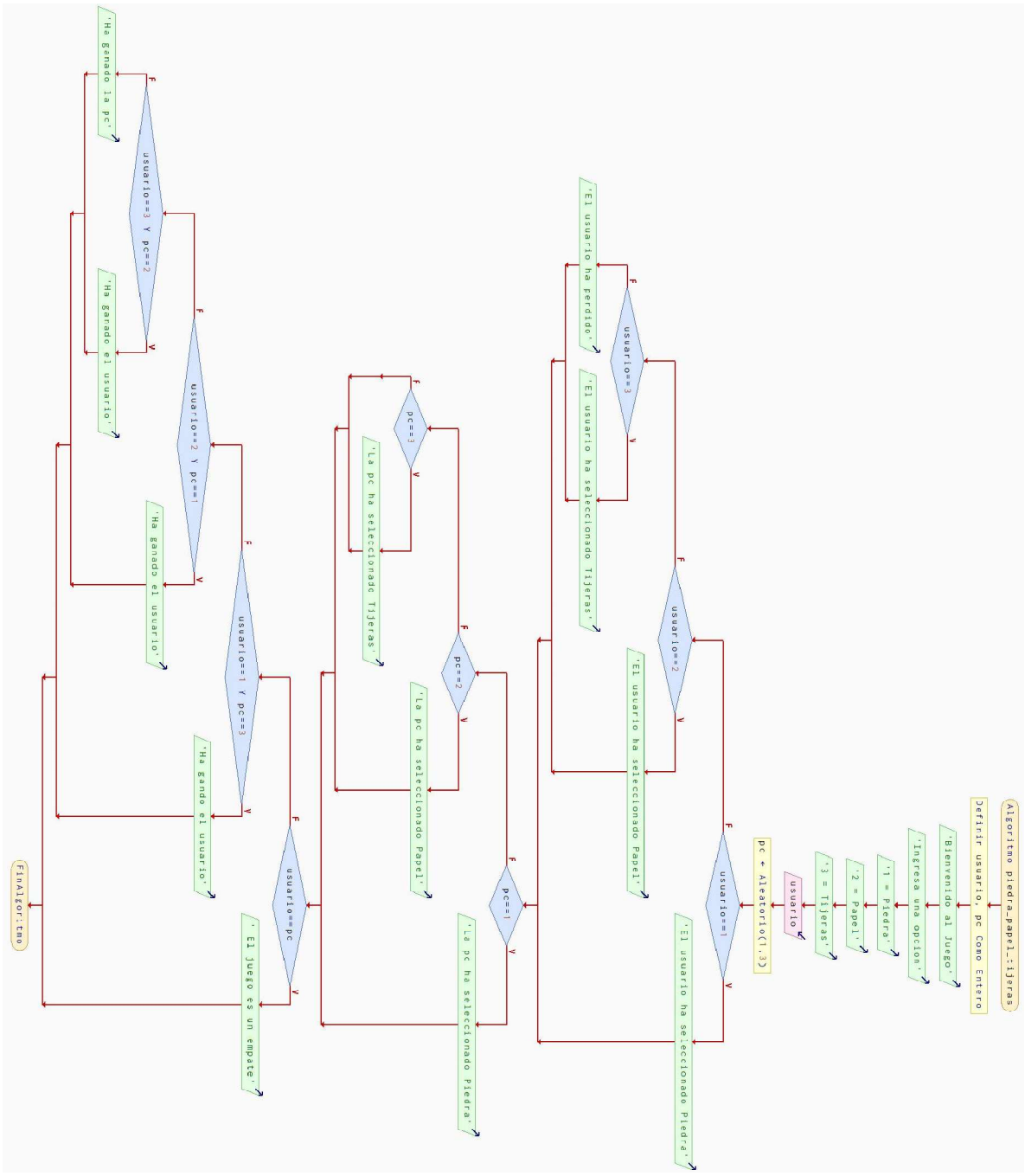
Uso de funciones

```
def jugar_ronda ():
    opciones = ["piedra", "papel", "tijeras"]
    usuario = input("Elige piedra, papel o tijeras: ").lower()

    while usuario not in opciones:
        usuario = input("Entrada invalida. Por favor ingresa piedra, papel o tijeras: ").lower()

    maquina = random.choice(opciones)

    print(f"Tú elegiste:  {usuario}")
    print(f"La máquina eligió: {maquina}")

    if usuario == maquina:
        print("Tenemos un Empate!!")
        return 0
    elif (usuario == "piedra" and maquina == "tijeras") or \
        (usuario == "papel" and maquina == "piedra") or \
        (usuario == "tijeras" and maquina == "papel"):
        print ("Eres el Ganador!!")
        return 1
```

8. Diagrama de Flujo realizo en PSEINT

Ilustración 18

Diagrama de flujo Juego Piedra, Papel y Tijera.



9. Conclusión

En la actualidad la lógica de programación es crucial porque se enfoca en el crecimiento de la resolución de problemas, permitiendo ejecutar tareas complejas en pasos claros y lógicos que las computadoras pueden ejecutar, esencial para todo, desde aplicaciones móviles, sitios webs, hasta la automatización industrial, impulsando la innovación, eficiencia y creación de nuevas oportunidades laborales en la era digital. Debemos tener presente que es el pedestal para escribir un código mantenible y eficiente, más allá de la construcción de un lenguaje específico. Es de gran importancia el poder desarrollar un entorno digital sostenible, inclusivo y responsable. Está en nosotros el saber aprovechar de forma responsable todos los beneficios que nos ofrece la era tecnológica. Al reflexionar sobre nuestro futuro tecnológico, es esencial priorizar la ética, la privacidad y el bienestar social para construir una sociedad más equitativa e inclusiva.

10. Bibliografía

Amazon Web Services. (2024). *¿En qué consiste la elaboración de diagramas de arquitecturas?* Obtenido de Amazon Web Services: https://aws.amazon.com/es/what-is/architecture-diagramming/

Amazon Web Services. (2025). *¿Qué es Python?* Obtenido de Amazon Web Services: https://aws.amazon.com/es/what-is/python/#:~:text=Los%20desarrolladores%20utilizan%20Python%20porque,aumenta%20la%20velocidad%20del%20desarrollo.

Busbee, K. L. (2018). *Programming Fundamentals*. Obtenido de Rebus Community: https://press.rebus.community/programmingfundamentals/chapter/flowcharts/

El Libro De Python. (2025). *Diccionarios*. Obtenido de El Libro De Python: https://ellibrodepython.com/diccionarios-en-python#diccionario

El Libro De Python. (2025). *Listas en Python*. Obtenido de El Libro De Python: https://ellibrodepython.com/listas-en-python#iterar-listas

El Libro De Python. (2025). *Tupla (tuple)*. Obtenido de El Libro De Python: https://ellibrodepython.com/tuplas-python

Martes Tecnológico. (6 de Septiembre de 2025). *El impacto de las nuevas tecnologías en la sociedad*. Obtenido de Martes Tecnológico: https://www.martestecnologico.com/el-impacto-de-las-nuevas-tecnologias-en-la-sociedad/

Visual Paradigm. (2025). *What is Use Case Diagram?* Obtenido de Visual Paradigm: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/