

**FACULTAD DE CIENCIAS E INGENIERÍA**

**CARRERA DE INGENIERIA DE SOFTWARE**

**MODALIDAD PRESENCIAL**

**ASIGNATURA:**

Técnica de programación

**DOCENTE:**

Víctor Hugo Cruz.

**ALUMNO:**

Frank Giancarlo Borja Zalamea.

**TEMA:**

Sentencias, Ciclos, Arreglos, Matrices en Python

**CURSO:**

SEGUNDO SEMESTRE PARALELO A-1

**PERIODO LECTIVO**

**2022-2023**

INDICE

[1.- Sentencia / Instrucciones en Python 3](#_Toc136468329)

[1.1.- ¿Qué es? 3](#_Toc136468330)

[1.2.- Objetivo de la Sentencias /Instrucciones en Python. 3](#_Toc136468331)

[1.3.- ¿Para qué se usan? 3](#_Toc136468332)

[1.4.- Tipos de Sentencias / Instrucciones 4](#_Toc136468333)

[1.4.- Sentencias de Python que no se utilizan en otros lenguajes de programación 6](#_Toc136468334)

[2.- Ciclos en Python 7](#_Toc136468335)

[2.1.- ¿Qué son? 7](#_Toc136468336)

[2.2.- Objetivo de los ciclos en Python 7](#_Toc136468337)

[2.3.- ¿Para qué se usan? 7](#_Toc136468338)

[2.4.- Tipos de Ciclos 7](#_Toc136468339)

[2.5.- Ciclos de Python que no se utilizan en otros lenguajes de programación 8](#_Toc136468340)

[3.- Arreglos en Python 8](#_Toc136468341)

[3.1.- ¿Qué es? 8](#_Toc136468342)

[3.2.- Objetivo de los arreglos 9](#_Toc136468343)

[3.3.- ¿Para qué se usan? 9](#_Toc136468344)

[3.4.- Tipos de Arreglos 10](#_Toc136468345)

[3.5.- Arreglos de Python que no se utilizan en otros lenguajes de programación 11](#_Toc136468346)

[4.- Matrices en Python 12](#_Toc136468347)

[4.1.- ¿Qué son? 12](#_Toc136468348)

[4.2.- Objetivo de las Matrices 12](#_Toc136468349)

[4.3.- ¿Para qué se usan? 13](#_Toc136468350)

[4.4.- Tipos de Matrices 14](#_Toc136468351)

[5.- Banco de preguntas. 14](#_Toc136468352)

[5.1.- Sentencias / Instrucciones en Python. 14](#_Toc136468353)

[5.2.- Ciclos Python. 16](#_Toc136468354)

[5.3.- Arreglos de Python. 19](#_Toc136468355)

[5.4.- Matrices en Python. 21](#_Toc136468356)

[Bibliografía 24](#_Toc136468357)

# 1.- Sentencia / Instrucciones en Python

## 1.1.- ¿Qué es?

En Python una sentencia en Python es una instrucción que el intérprete del lenguaje puede llevar a cabo y también son que las instrucciones consisten en una combinación de palabras clave, operadores, valores, variables y llamadas a funciones, y deben ajustarse a la estructura sintáctica del lenguaje. Python instrucciones ofrece diversos tipos de instrucciones, incluyendo instrucciones de solicitud, instrucciones de impresión, instrucciones condicionales (if-else) y bucles (for, while), entre otros ejemplos. Estas instrucciones son fundamentales para desarrollar programas eficientes que involucren lógica, iteración y manipulación de datos.

## 1.2.- Objetivo de la Sentencias /Instrucciones en Python.

El objetivo principal de las sentencias en Python es otorgar el poder de controlar la forma en que un programa se ejecuta. Esto implica tomar decisiones según ciertas condiciones, repetir bloques de código en bucles y manejar excepciones de manera adecuada.

También podemos decir que el propósito de las sentencias en Python es gestionar el orden de ejecución de un programa, permitiendo establecer bloques de código que se activan según determinadas condiciones o para llevar a cabo tareas específicas en una secuencia específica.   
Las sentencias en Python desempeñan un papel esencial al determinar el comportamiento y la funcionalidad de un programa, ya que permiten ejecutarlo de manera secuencial y manipular los datos según las necesidades específicas del proyecto en cuestión.

## 1.3.- ¿Para qué se usan?

Las sentencias en Python nos brindan la capacidad de controlar el flujo de nuestro programa, permitiéndonos determinar si ciertas líneas de código deben ejecutarse en función del cumplimiento de condiciones predefinidas. Específicamente, se encargan de evaluar si una condición es verdadera o falsa para tomar decisiones sobre qué acciones realizar.

Otro uso importante de las sentencias en Python es la capacidad de manipular y transformar datos. Estas sentencias permiten llevar a cabo operaciones aritméticas, asignar valores a variables, concatenar cadenas de texto, convertir tipos de datos y realizar diversas manipulaciones en los datos según sea necesario. Esto brinda flexibilidad para trabajar con información y realizar operaciones específicas en un programa.

## 1.4.- Tipos de Sentencias / Instrucciones

1. **Sentencia de asignación:** Se emplea para asignar un valor a una variable, evaluando una expresión en el lado derecho del operador "=" y asignando el resultado a la variable en el lado izquierdo.
2. **Sentencia de impresión:** Utilizada para mostrar valores en la consola, la sentencia PRINT envía datos a la pantalla, a una impresora de líneas u otro archivo de impresión.
3. **Sentencia "if":** Esta sentencia permite ejecutar código Python si una condición se cumple. Se emplea la palabra clave "if" seguida de la condición que debe satisfacerse. El programador también puede definir la acción que se llevará a cabo si se cumple la condición.
4. **Sentencia "elif":** Se pueden agregar un número ilimitado de condiciones adicionales a una sentencia "if-else" utilizando la sentencia "elif", que es una abreviatura de "else if".
5. **Sentencia "else":** Opcionalmente, se puede agregar una sentencia "else" a la sentencia "if". Esta sentencia "else" contiene el código que se ejecuta cuando no se cumple la condición especificada en la sentencia "if".
6. **Sentencia de bucle "while":** Se emplea para repetir un bloque de código mientras se cumpla una condición específica. La sentencia "while" en Python es una estructura de control de flujo utilizada para ejecutar repetidamente un bloque de instrucciones mientras se cumpla una condición determinada.
7. **Sentencia de bucle "for":** Se utiliza para iterar sobre una secuencia de elementos. En cada iteración, la variable "elemento" adquiere el valor del siguiente elemento en la secuencia, y se ejecuta el bloque de código indentado bajo la declaración "for" para ese elemento. Este proceso se repite para cada elemento en la secuencia hasta que todos los elementos han sido procesados.
8. **Sentencia "pass":** Es una operación nula que no realiza ninguna acción cuando se ejecuta. Se utiliza cuando se requiere una declaración sintáctica pero no se desea ejecutar ningún comando o código.
9. **Sentencia break:** La sentencia "break" en Python se utiliza dentro de un bucle, ya sea "for" o "while", para interrumpir su ejecución antes de que se complete su ciclo normal. Cuando se encuentra la instrucción "break", el programa sale inmediatamente del bucle y continúa con la ejecución del código que sigue después del bucle. Esta sentencia es especialmente útil cuando se desea detener un bucle prematuramente basándose en una condición específica. Al usar "break", se puede evitar que el bucle continúe ejecutándose innecesariamente una vez que se haya cumplido cierta condición, lo que permite un mayor control y eficiencia en el flujo del programa.
10. **Sentencia def:** La instrucción "def" en Python se emplea para crear y definir funciones. Al utilizar la palabra clave "def", seguida del nombre de la función y paréntesis, se establece un bloque de código que incluye la lógica y las operaciones que se ejecutarán cuando se invoque la función. Esta instrucción es fundamental en la programación en Python, ya que facilita la modularización del código y la reutilización de funciones en distintas partes de un programa. Al definir una función con "def", se crea un punto de entrada donde se especifican las operaciones que se llevarán a cabo al llamar a la función con los argumentos adecuados.
11. **Sentencia with:** Se utiliza para simplificar la gestión de recursos, como archivos o conexiones de red, garantizando que se liberen adecuadamente al finalizar el bloque.
12. **Sentencia match:** La sentencia "match" se utiliza para realizar la coincidencia de patrones.
13. **Sentencia except:** La sentencia "except" en Python se utiliza en conjunto con la sentencia "try" para capturar y manejar excepciones específicas. Al colocar el bloque de código que podría generar una excepción dentro de un bloque "try", se tiene la oportunidad de capturar y manejar dicha excepción utilizando la sentencia "except".
14. **Sentencia try:** La sentencia "try" en Python se utiliza para manejar excepciones y errores en el código. Al utilizar la estructura "try" seguida de un bloque de código, se establece un contexto en el que se monitorean posibles excepciones.
15. **Sentencia finally:** La sentencia "finally" en Python se emplea junto con las sentencias "try" y "except" para establecer un bloque de código que será ejecutado sin importar si se produce una excepción o no. El bloque "finally" se coloca después de los bloques "try" y "except" y contiene las instrucciones que deben ejecutarse sin importar el resultado del bloque "try".

## 1.4.- Sentencias de Python que no se utilizan en otros lenguajes de programación

**Comprensión de listas (List comprehensions):** La comprensión de listas en Python es una característica que proporciona una forma breve y expresiva de crear listas utilizando una secuencia o estructura de datos previa. Esta funcionalidad permite generar listas de manera eficiente y en una sola línea de código, lo cual resulta especialmente útil para llevar a cabo operaciones como filtrado, mapeo y transformación de datos en forma concisa.

**Generadores (Generators):** Los generadores en Python son una solución eficiente para recorrer conjuntos de datos extensos sin tener que almacenarlos completamente en memoria. En lugar de generar una lista completa de valores, los generadores producen valores de forma perezosa, es decir, uno a la vez, a medida que se requieren. Esto permite un uso más eficiente de los recursos, ya que los valores se generan bajo demanda, evitando así la carga innecesaria de datos en memoria.

**Desempaquetado de tuplas (Tuple unpacking):** El desempaquetado de tuplas en Python es una característica que permite asignar múltiples valores a varias variables al mismo tiempo. Esta funcionalidad permite asignar de forma rápida y sencilla los elementos de una tupla a variables individuales. Es una forma eficiente y conveniente de extraer y asignar valores de una tupla en una sola línea de código.

**Gestión de excepciones (Exception handling):** La gestión de excepciones en Python se refiere al sólido sistema integrado que permite detectar y manejar errores durante la ejecución de un programa. Esto proporciona una forma estructurada de responder a situaciones excepcionales y manejar errores de manera adecuada. Gracias a este sistema, es posible escribir código más robusto al anticiparse y gestionar posibles fallos. Python ofrece una variedad de mecanismos y palabras clave para capturar, manejar y controlar excepciones, lo que contribuye a mejorar la confiabilidad y el flujo de ejecución del programa.

# 2.- Ciclos en Python

## 2.1.- ¿Qué son?

Los ciclos o bucles, también denominados estructuras de control repetitivas, desempeñan un papel crucial en el desarrollo de programas. Estas construcciones en Python te permiten repetir una o varias instrucciones tantas veces como sea necesario, brindando flexibilidad y eficiencia en la ejecución de tareas, también podemos decir que son estructuras secuenciales que nos permiten ejecutar una serie de operaciones de forma repetida sobre valores específicos que cambian en cada iteración del ciclo.

## 2.2.- Objetivo de los ciclos en Python

El propósito fundamental de los ciclos en Python es posibilitar la ejecución repetida y controlada de uno o varios conjuntos de instrucciones. Estos ciclos se utilizan cuando es necesario repetir un bloque de código múltiples veces, ya sea un número específico de repeticiones o hasta que se cumpla una determinada condición. Esto permite automatizar tareas que requieren repetición y procesar conjuntos de datos de manera eficiente.

En sí, el objetivo principal de los ciclos en Python es optimizar el flujo de ejecución del programa al repetir instrucciones de forma iterativa y controlada.

## 2.3.- ¿Para qué se usan?

Los bucles son utilizados en los programas para implementar iteraciones, lo cual implica ejecutar un bloque de código repetidamente mientras se cumple una condición específica. Cuando la condición se vuelve falsa, el programa sale del bucle y continúa con la ejecución secuencial. En resumen, los bucles permiten repetir un conjunto de instrucciones hasta que se alcance una condición de salida, lo que facilita la automatización de tareas repetitivas en un programa.

## 2.4.- Tipos de Ciclos

**Bucle while:** Los ciclos "while" en Python son aquellos que se ejecutan mientras una condición específica sea verdadera (True). Primero se verifica si la condición se cumple y, en caso afirmativo, se ejecuta el bloque de código correspondiente. Esto se repite continuamente hasta que la condición ya no se cumpla, momento en el que el ciclo "while" se detiene y la ejecución del programa continúa con las instrucciones siguientes. En resumen, los ciclos "while" permiten ejecutar repetidamente un bloque de código mientras se cumpla una condición determinada.

**Bucle For:** Los ciclos "for" en Python, a diferencia de los ciclos "while", tienen una duración predefinida y una sintaxis diferente que nos permite utilizar una variable temporal que se actualizará en cada iteración. Esto nos facilita recorrer secuencias de elementos, como listas o rangos, ejecutando un bloque de código para cada valor de la variable temporal. En resumen, los ciclos "for" nos permiten iterar de forma controlada sobre una secuencia de elementos con una variable temporal que se actualiza en cada iteración.

## 2.5.- Ciclos de Python que no se utilizan en otros lenguajes de programación

**Ciclo "for" con "else":** En Python, se puede añadir una cláusula "else" al final de un ciclo "for". Esta cláusula se activa cuando el ciclo ha recorrido todos los elementos de la secuencia sin interrupciones, es decir, si no se ha ejecutado la instrucción "break".

**Ciclo "while" con "else":** En Python, también es posible agregar una cláusula "else" al final de un ciclo "while". Esta cláusula se ejecuta cuando la condición del ciclo se vuelve falsa y la ejecución del ciclo se completa de forma normal, sin ninguna interrupción por una instrucción "break".

**Ciclo "itertools":** El ciclo "itertools" en Python se refiere al módulo "itertools" que proporciona diversas herramientas para la iteración eficiente sobre secuencias en Python. No es propiamente un ciclo en sí, pero se utiliza en conjunto con ciclos como "for" para generar iteraciones especiales o realizar operaciones combinadas en los elementos de una secuencia.

# 3.- Arreglos en Python

## 3.1.- ¿Qué es?

Un arreglo en Python es un tipo de variable especial que puede almacenar múltiples valores al mismo tiempo. En lugar de tener los datos almacenados en un solo lugar, el arreglo consta de una serie de contenedores donde se guardan diferentes datos en cada uno de ellos. Cada contenedor del arreglo representa una ubicación separada donde se almacena un valor específico.

Los arreglos en Python son flexibles y dinámicos, lo que significa que se pueden cambiar en tamaño y contenido durante la ejecución del programa. Para acceder a los elementos del arreglo, se utilizan índices numéricos que indican su posición relativa. Los arreglos son ampliamente utilizados en Python para almacenar colecciones de datos relacionados, lo que facilita la manipulación y procesamiento de la información en los programas.

## 3.2.- Objetivo de los arreglos

El objetivo principal de los arreglos en Python es brindar una estructura de datos eficiente y práctica para almacenar y manipular colecciones de elementos relacionados. Al utilizar arreglos, podemos organizar los datos de manera secuencial y acceder a ellos mediante índices, lo que facilita la búsqueda, modificación y eliminación de elementos específicos. Los arreglos también nos permiten realizar operaciones en conjunto, como ordenar, filtrar y realizar cálculos estadísticos en los datos almacenados. En resumen, el objetivo fundamental de los arreglos en Python es proporcionar una herramienta versátil y efectiva para trabajar con conjuntos de datos organizados, simplificando así el proceso de manipulación y análisis de información.

## 3.3.- ¿Para qué se usan?

Los arreglos en Python se emplean para almacenar y organizar colecciones de elementos relacionados. Sus aplicaciones y usos principales son los siguientes:

* **Almacenamiento de datos:** Los arreglos permiten guardar datos de distintos tipos en una estructura secuencial, lo que facilita su posterior acceso y manipulación.
* **Búsqueda y acceso de elementos:** Los arreglos se utilizan para acceder a elementos específicos mediante índices, lo cual agiliza la búsqueda y recuperación de datos en una colección.
* **Modificación y actualización de datos:** Los arreglos posibilitan modificar los valores de los elementos almacenados, lo cual resulta especialmente útil al requerir actualizar o cambiar información en una colección.

## 3.4.- Tipos de Arreglos

**Arrays unidimensionales:** En Python, los arreglos unidimensionales se implementan utilizando listas. Las listas en Python son capaces de almacenar elementos de diferentes tipos, y se puede acceder a cada elemento individual mediante un índice que indica su posición dentro de la lista. Esta característica permite la manipulación y el acceso individual a los elementos con base en su posición en la lista. En resumen, las listas en Python proporcionan una forma conveniente de representar y trabajar con arreglos unidimensionales al permitirnos almacenar y acceder a elementos de diferentes tipos mediante el uso de índices.

**Arrays multidimensionales:** En Python, los arreglos multidimensionales se pueden implementar utilizando diferentes estructuras de datos, como listas anidadas o utilizando la biblioteca NumPy.

**Listas**: En Python, las listas son una estructura de datos proporcionada por el lenguaje y se consideran un tipo de arreglo. Son esenciales en la programación, ya que nos permiten abordar una amplia gama de problemas que serían difíciles de resolver sin ellas. En Python, las listas son especialmente valiosas, ya que nos permiten almacenar múltiples valores en una sola variable, lo que simplifica la manipulación y organización de los datos.

**Tuplas:** Las tuplas en Python, conocidas como "tuples", tienen similitudes con las listas, pero se distinguen por dos diferencias importantes. En primer lugar, las tuplas son estructuras de datos inmutables, lo que implica que no pueden modificarse una vez que se han creado.

**Arrays de la biblioteca array:** La biblioteca "array" de Python proporciona un tipo de dato llamado "array", el cual es similar a las listas en cuanto a su estructura, pero se diferencia en que es homogéneo, lo que significa que todos sus elementos deben ser del mismo tipo. Esta característica permite un uso más eficiente de la memoria y un acceso más rápido a los datos almacenados en el array. A diferencia de las listas, que pueden contener elementos de diferentes tipos, los arrays de la biblioteca "array" se utilizan cuando se requiere almacenar datos del mismo tipo de manera eficiente.

**La librería Numpy:** NumPy es una biblioteca de Python que se ha creado con el propósito específico de facilitar el cálculo numérico y el análisis de datos, especialmente en el contexto de conjuntos de datos extensos. Esta biblioteca introduce una nueva clase de objetos llamados "arrays" que permiten representar colecciones de datos del mismo tipo en múltiples dimensiones. Además, NumPy proporciona funciones altamente eficientes que facilitan la manipulación y operaciones sobre estos arrays. En resumen, NumPy se utiliza para simplificar y optimizar el manejo de datos numéricos y el análisis estadístico en Python.

## 3.5.- Arreglos de Python que no se utilizan en otros lenguajes de programación

**Listas por comprensión (List Comprehension):** Las listas por comprensión en Python son una forma concisa y poderosa de crear listas utilizando una sintaxis especial. Permiten generar listas de manera eficiente basándose en iteraciones o filtrado de elementos de otra lista, evitando la necesidad de utilizar bucles tradicionales.

**Generadores (Generators):** Los generadores en Python son una característica única que permite crear iteradores de forma eficiente. A diferencia de las listas por comprensión, que generan y almacenan todos los elementos en memoria de una vez, los generadores generan los elementos de manera incremental, según se necesiten. Esto resulta en un ahorro de espacio en memoria y un mejor rendimiento, ya que no es necesario generar y almacenar todos los elementos de antemano.

**Slicing:** Python proporciona una característica llamada "slicing" que permite acceder a subconjuntos de elementos en una lista. Con el slicing, es posible especificar rangos de índices para obtener una porción específica de la lista.

**Comprobación de pertenencia:** En Python, se utiliza el operador "in" para verificar la pertenencia de un elemento en una lista. Esta funcionalidad permite realizar comprobaciones de pertenencia de manera simple y clara, lo cual resulta muy útil al buscar elementos específicos o filtrar listas.

El operador "in" se utiliza en combinación con una lista y devuelve un valor booleano (True o False) que indica si el elemento se encuentra presente en la lista. Por ejemplo, si tenemos una lista llamada "lista" y queremos verificar si el elemento "x" está presente en ella, podemos escribir "x in lista". Si la condición es verdadera, es decir, si "x" se encuentra en la lista, el resultado será True; de lo contrario, será False.

# 4.- Matrices en Python

## 4.1.- ¿Qué son?

Cuando nos referimos a matrices en Python, nos estamos refiriendo a una estructura de datos bidimensional rectangular especializada que almacena datos organizados en filas y columnas. Estos datos pueden ser números, cadenas de texto, símbolos, expresiones y más. Las matrices son una de las estructuras de datos fundamentales utilizadas en cálculos matemáticos y científicos. Proporcionan una forma conveniente de organizar y manipular datos en un formato tabular, lo que facilita realizar operaciones matemáticas y análisis numérico en conjuntos de datos.

En sí, las matrices en Python son una herramienta esencial para realizar cálculos y análisis científicos al permitir el almacenamiento y manipulación de datos en un formato bidimensional.

## 4.2.- Objetivo de las Matrices

El objetivo principal de las matrices en Python es ofrecer una estructura de datos eficiente y práctica para almacenar y manipular información en forma de una cuadrícula bidimensional compuesta por filas y columnas. Algunos de los objetivos clave de las matrices en Python incluyen:

* Organización estructurada: Las matrices permiten organizar datos de manera ordenada en una estructura bidimensional, lo que facilita la gestión y el acceso a los elementos según su posición en la cuadrícula.
* **Operaciones matemáticas y científicas:** Las matrices son ampliamente utilizadas en cálculos matemáticos y científicos, ya que proporcionan una representación eficiente de datos que pueden ser sometidos a operaciones matriciales como multiplicación, suma, inversión, entre otras.
* **Análisis de datos estructurados:** Las matrices son útiles en el análisis de datos, ya que permiten almacenar y manipular conjuntos de datos estructurados en filas y columnas. Esto facilita la realización de operaciones estadísticas, filtrado de datos y cálculos relacionados con el análisis de datos.
* **Representación gráfica:** Las matrices se utilizan comúnmente para representar información gráfica como imágenes y gráficos. Cada elemento de la matriz puede representar un píxel de una imagen o un punto en un gráfico, lo que permite realizar operaciones de procesamiento de imágenes y generación de gráficos.

En sí, el objetivo principal de las matrices en Python es proporcionar una estructura de datos versátil y eficiente para almacenar, manipular y analizar datos estructurados en forma de una cuadrícula bidimensional, siendo especialmente útiles en cálculos matemáticos, análisis de datos y representación gráfica.

## 4.3.- ¿Para qué se usan?

Las matrices en Python tienen una amplia gama de aplicaciones, algunas de las cuales incluyen:

* **Análisis de datos:** Las matrices son utilizadas en el análisis de datos para organizar y manipular conjuntos de datos estructurados en una cuadrícula bidimensional. Esto permite realizar cálculos estadísticos, realizar operaciones matriciales y extraer información relevante de los datos.
* **Procesamiento de imágenes y gráficos:** Las matrices son especialmente útiles en el procesamiento de imágenes y gráficos, ya que cada píxel de una imagen o punto en un gráfico puede ser representado por un elemento en la matriz. Esto permite realizar operaciones de filtrado, transformación y manipulación de imágenes, así como la generación y visualización de gráficos.
* **Modelado matemático y científico:** Las matrices son fundamentales en el modelado matemático y científico, ya que permiten representar sistemas de ecuaciones lineales y realizar cálculos matriciales para resolver problemas complejos en áreas como la física, la ingeniería y la economía.
* **Aprendizaje automático y análisis numérico:** Las matrices son utilizadas en algoritmos de aprendizaje automático y análisis numérico, ya que proporcionan una forma eficiente de representar y manipular datos en forma de matrices. Esto facilita el entrenamiento de modelos de aprendizaje automático, la realización de cálculos numéricos y la implementación de algoritmos de optimización.
* **Simulación y modelado computacional:** Las matrices son esenciales en la simulación y el modelado computacional, ya que permiten representar sistemas complejos en forma de matrices y realizar operaciones matriciales para simular el comportamiento y analizar el rendimiento de estos sistemas.

## 4.4.- Tipos de Matrices

* **Matrices dispersas (Sparse Matrices):** Las matrices dispersas son matrices en las que la mayoría de los elementos son nulos o ceros. En Python, la biblioteca SciPy ofrece una estructura de datos llamada "scipy. sparse" que permite trabajar de manera eficiente con este tipo de matrices.
* **Matrices densas (Dense Matrices):** Las matrices densas son matrices en las que la mayoría de los elementos no son nulos. En Python, tanto las matrices de NumPy como las listas anidadas son comúnmente utilizadas para representar matrices densas.
* **Matrices cuadradas:** Las matrices cuadradas son matrices en las que el número de filas es igual al número de columnas. Estas matrices son particularmente relevantes en el álgebra lineal y se utilizan para representar transformaciones lineales y sistemas de ecuaciones.
* **Matrices diagonales:** Las matrices diagonales son matrices cuadradas en las que todos los elementos fuera de la diagonal principal son nulos. Estas matrices se utilizan en diversas aplicaciones, como el análisis de sistemas lineales y la representación compacta de matrices dispersas.

# 5.- Banco de preguntas.

## 5.1.- Sentencias / Instrucciones en Python.

1. ¿Cuál es el objetivo principal de las sentencias en Python?
2. Evaluar expresiones aritméticas.
3. Controlar el flujo de ejecución de un programa.
4. Crear estructuras de datos complejas.
5. Realizar operaciones matemáticas complejas.

Respuesta correcta: **b)**

1. ¿Cuál es el uso principal de la sentencia "if" en Python?
2. Asignar un valor a una variable.
3. Mostrar valores en la consola.
4. Repetir un bloque de código mientras se cumpla una condición.
5. Ejecutar código si una condición se cumple.

Respuesta correcta: **d)**

1. ¿Qué tipo de sentencia se utiliza para iterar sobre una secuencia de elementos en Python?
2. Sentencia de asignación.
3. Sentencia de impresión.
4. Sentencia "if".
5. Sentencia de bucle "for".

Respuesta correcta: **d)**

1. ¿Cuál es el propósito de la sentencia "pass" en Python?
2. Asignar un valor a una variable.
3. Mostrar valores en la consola.
4. Realizar operaciones aritméticas.
5. No realizar ninguna acción.

Respuesta correcta: **d)**

1. ¿Cuál es el uso principal de la sentencia "break" en Python?
2. Asignar un valor a una variable.
3. Mostrar valores en la consola.
4. Interrumpir la ejecución de un bucle.
5. Ejecutar código si una condición se cumple.

Respuesta correcta: **c)**

1. ¿Cuál es el objetivo principal de la sentencia "def" en Python?
2. Asignar un valor a una variable.
3. Mostrar valores en la consola.
4. Crear y definir funciones.
5. Realizar operaciones aritméticas.

Respuesta correcta: **c)**

1. ¿Cuál es el uso principal de la sentencia "with" en Python?
2. Simplificar la gestión de recursos.
3. Mostrar valores en la consola.
4. Evaluar expresiones aritméticas.
5. Realizar operaciones de comparación.

Respuesta correcta: **a)**

1. ¿Qué tipo de sentencia se utiliza para capturar y manejar excepciones en Python?
2. Sentencia "if".
3. Sentencia "with".
4. cSentencia "try".
5. Sentencia "def".

Respuesta correcta: **c)**

1. ¿Cuál es el propósito de la sentencia "finally" en Python?
2. Asignar un valor a una variable.
3. Mostrar valores en la consola.
4. Establecer un bloque de código que se ejecuta sin importar si se produce una excepción o no.
5. Realizar operaciones matemáticas complejas.

Respuesta correcta: **c)**

1. ¿Qué característica de Python permite crear listas de forma concisa utilizando una secuencia o estructura de datos previa?
2. Comprensión de listas.
3. Generadores.
4. Desempaquetado de tuplas.
5. Gestión de excepciones.

Respuesta correcta: **a).**

## 5.2.- Ciclos Python.

1. ¿Cuál es el propósito de los ciclos en Python?
2. Realizar operaciones matemáticas.
3. Automatizar tareas repetitivas.
4. Imprimir mensajes en la consola.
5. Validar entradas de usuario.

Respuesta correcta: **b)**

1. ¿Qué son los ciclos en Python?
2. Estructuras de control para decisiones.
3. Estructuras de control para iteraciones.
4. Estructuras de datos para almacenamiento.
5. Estructuras para definir funciones.

Respuesta correcta: **b)**

1. ¿Cuál de los siguientes ciclos se ejecuta mientras una condición sea verdadera?
2. Ciclo if
3. Ciclo for
4. Ciclo while
5. Ciclo do-while

Respuesta correcta: **c)**

1. ¿Qué tipo de ciclo en Python recorre una secuencia de elementos?
2. Ciclo for
3. Ciclo while
4. Ciclo do-while
5. Ciclo foreach

Respuesta correcta: **a)**

1. ¿Cuál de los siguientes ciclos tiene una duración predefinida en Python?
2. Ciclo if
3. Ciclo for
4. Ciclo while
5. Ciclo do-while

Respuesta correcta: **b)**

1. ¿Qué sucede al utilizar la instrucción "break" dentro de un ciclo?
2. El ciclo se detiene y el programa continúa con la siguiente instrucción.
3. El ciclo se repite infinitamente.
4. El ciclo cambia a la siguiente iteración sin ejecutar el resto del bloque.
5. El ciclo se ejecuta solo una vez.

Respuesta correcta: **a)**

1. ¿Qué cláusula se puede añadir al final de un ciclo "for" en Python?
2. if
3. else
4. while
5. for-each

Respuesta correcta: **b)**

1. ¿Qué cláusula se puede añadir al final de un ciclo "while" en Python?
2. if
3. else
4. for
5. do-while

Respuesta correcta: **b)**

1. ¿Cuál de los siguientes módulos se utiliza para la iteración eficiente sobre secuencias en Python?
2. itertools
3. math
4. random
5. d) os

Respuesta correcta: **a)**

1. ¿Cuál de los siguientes ciclos no se utiliza en Python?
2. Ciclo for
3. Ciclo while
4. Ciclo do-while
5. Ciclo repeat-until

Respuesta correcta: **d)**

## 5.3.- Arreglos de Python.

1. ¿Qué es un arreglo en Python?
2. Una estructura de datos que solo puede almacenar números enteros.
3. Un tipo de variable que puede almacenar múltiples valores al mismo tiempo.
4. Una función incorporada en Python para realizar operaciones matemáticas.
5. Un bucle utilizado para repetir un bloque de código.

Respuesta correcta**: b)**

1. ¿Cuál es el objetivo principal de los arreglos en Python?
2. Almacenar solo valores numéricos.
3. Simplificar la manipulación y análisis de datos.
4. Realizar cálculos matemáticos complejos.
5. Mejorar el rendimiento de los bucles.

Respuesta correcta: **b)**

1. ¿Para qué se utilizan principalmente los arreglos en Python?
2. Almacenar y organizar colecciones de elementos relacionados.
3. Realizar operaciones matemáticas complejas.
4. Controlar el flujo de ejecución de un programa.
5. Validar entradas de usuario.

Respuesta correcta: **a)**

1. ¿Cuál de los siguientes tipos de arreglos en Python permite almacenar elementos de diferentes tipos en una estructura secuencial?
2. Arrays unidimensionales.
3. Arrays multidimensionales.
4. Listas.
5. Tuplas.

Respuesta correcta: **c)**

1. ¿Qué característica distingue a las tuplas de las listas en Python?
2. Las tuplas son estructuras de datos inmutables.
3. Las tuplas solo pueden almacenar elementos numéricos.
4. Las tuplas permiten modificar los valores de los elementos almacenados.
5. Las tuplas tienen una longitud fija y no se pueden cambiar.

Respuesta correcta: **a)**

1. ¿Qué biblioteca de Python se utiliza para trabajar con arreglos multidimensionales de manera eficiente?
2. array
3. numpy
4. pandas
5. matplotlib

Respuesta correcta: **b)**

1. ¿Qué tipo de estructura de datos se utiliza en Python para crear listas de manera eficiente basándose en iteraciones o filtrado de elementos de otra lista?
2. Listas por comprensión (List Comprehension).
3. Generadores (Generators).
4. Slicing.
5. Comprobación de pertenencia.

Respuesta correcta: **a)**

1. ¿Cuál de las siguientes afirmaciones sobre los generadores en Python es correcta?
2. Los generadores generan y almacenan todos los elementos en memoria de una vez.
3. Los generadores son estructuras de datos inmutables.
4. Los generadores generan los elementos de manera incremental, según se necesiten.
5. Los generadores solo pueden almacenar valores numéricos.

Respuesta correcta: **c)**

1. ¿Qué característica de los arreglos en Python permite acceder a subconjuntos de elementos en una lista?
2. Listas por comprensión (List Comprehension).
3. Generadores (Generators).
4. Slicing.
5. Comprobación de pertenencia.

Respuesta correcta: **c)**

1. ¿Qué operador se utiliza en Python para verificar la pertenencia de un elemento en una lista?
2. +
3. \*
4. /
5. in

Respuesta correcta: **d)**

## 5.4.- Matrices en Python.

1. ¿Qué son las matrices en Python?
2. Estructuras de datos unidimensionales para almacenar elementos relacionados.
3. Estructuras de datos bidimensionales para almacenar datos organizados en filas y columnas.
4. Funciones incorporadas en Python para realizar operaciones matemáticas.
5. Bucles utilizados para repetir un bloque de código.

Respuesta correcta**: b)**

1. ¿Cuál es el objetivo principal de las matrices en Python?
2. Almacenar datos en una estructura bidimensional ordenada.
3. Realizar cálculos matemáticos y científicos.
4. Facilitar el análisis de datos estructurados.
5. Representar gráficamente información como imágenes y gráficos.

Respuesta correcta: a) Almacenar datos en una estructura bidimensional ordenada.

1. ¿Para qué se utilizan principalmente las matrices en Python?
2. Análisis de datos y cálculos matemáticos.
3. Procesamiento de imágenes y gráficos.
4. Modelado matemático y científico.
5. Todas las anteriores.

Respuesta correcta: d) Todas las anteriores.

1. ¿Cuál es el tipo de matriz en la que la mayoría de los elementos son nulos o ceros?
2. Matrices dispersas.
3. Matrices densas.
4. Matrices cuadradas.
5. Matrices diagonales.

Respuesta correcta: a) Matrices dispersas.

1. ¿Qué biblioteca de Python se utiliza para trabajar de manera eficiente con matrices dispersas?
2. NumPy.
3. Pandas.
4. SciPy.
5. Matplotlib.

Respuesta correcta: c) SciPy.

1. ¿Cuál es el tipo de matriz en la que la mayoría de los elementos no son nulos?
2. Matrices dispersas.
3. Matrices densas.
4. Matrices cuadradas.
5. Matrices diagonales.

Respuesta correcta: b) Matrices densas.

1. ¿Cuál es el tipo de matriz en la que el número de filas es igual al número de columnas?
2. Matrices dispersas.
3. Matrices densas.
4. Matrices cuadradas.
5. Matrices diagonales.

Respuesta correcta: c) Matrices cuadradas.

1. ¿Qué tipo de matrices se utilizan para representar transformaciones lineales y sistemas de ecuaciones?
2. Matrices dispersas.
3. Matrices densas.
4. Matrices cuadradas.
5. Matrices diagonales.

Respuesta correcta: c) Matrices cuadradas.

1. ¿Cuál es el tipo de matriz en la que todos los elementos fuera de la diagonal principal son nulos?
2. Matrices dispersas.
3. Matrices densas.
4. Matrices cuadradas.
5. Matrices diagonales.

Respuesta correcta: d) Matrices diagonales.

1. ¿Cuál es la biblioteca de Python ampliamente utilizada para trabajar con matrices en general?
2. NumPy.
3. SciPy.
4. Pandas.
5. Matplotlib.

Respuesta correcta: a) NumPy.

# Bibliografía

Industrial, E. I. *Sentencias de definición, Inicialización y asignación*. Sentencias de definición, inicialización y asignación - Fundamentos de Programación en C++. https://acortar.link/NG7DNp

KeepCoding, R. (2023, April 11). *¿Qué son los bucles o Ciclos de Programación?*. KeepCoding Bootcamps. https://acortar.link/zhmwQX

If ... ELIF ... else *...* if ... elif ... else ... Python. Bartolomé Sintes Marco. www.mclibre.org. (2022, February 13). https://www.mclibre.org/consultar/python/lecciones/python-if-else.html

Saavedra, J. A. (2023a, May 3). Ciclos en python: Qué son, cómo funcionan los bucles for y while y cómo USARLOS. Ebac. <https://ebac.mx/blog/ciclos-en-python>

Programación en Castellano, S. L. (2021, March 24). Cómo funcionan las sentencias condicionales en Python. Programación en Castellano. https://programacion.net/articulo/como\_funcionan\_las\_sentencias\_condicionales\_en\_python\_1501

Llerena Izquierdo, J. (2020a, August 8). Codifica en Python. Repositorio Institucional de la Universidad Politécnica Salesiana: Página de inicio. https://dspace.ups.edu.ec/handle/123456789/19346