**1.Introducción**

Python es un lenguaje de programación interpretado, funcional, orientado a objetos, interactivo y multiparadigmático donde se permite la creación de módulos y librerías. Es caracterizado por su alta legibilidad y comprensión de código, haciéndolo más sencillo tanto para el programador, como el que lee el código de forma externa. Su estructura permite características tales como un manejo adecuado de excepciones y tipos de datos de alto nivel[1].

Las listas en Python son estructuras de datos secuenciales, que vienen incluidas con el core

del lenguaje, y que permiten almacenar en su interior referencias a objetos [2]. Las listas son una estructura de datos muy versátil y útil para cualquier programador, ya que permiten almacenar múltiples elementos de distintos tipos en una sola variable. En este informe se presentarán algunos de los métodos más destacados que ofrecen las listas en Python, con énfasis en su aplicación en objetos pertenecientes a clases definidas por el programador.[2]

Palabras Claves: Python, Lista

**2. Desarrollo**

**2.1 Ordenamiento de listas**

Las listas en Python ofrecen varios métodos de ordenamiento que permiten reorganizar los elementos de la lista de manera ascendente o descendente según el tipo de dato que contengan. Dos de los métodos más utilizados son sort() y sorted().[4][5]

El método sort() ordena la lista original en orden ascendente, y no retorna ningún valor[4][5]:

*#Se crea una lista desordenada*

*lista = [5, 2, 8, 1, 3]*

*#Se utiliza el método de ordenamiento ascendente*

*lista.sort()*

*print(lista)*

*# Salida: [1, 2, 3, 5, 8]*

El método sorted() retorna una nueva lista ordenada según los elementos de la lista original, y permite ordenar también en orden descendente con el argumento opcional reverse=True[4][5]:

*#Se crea una lista de números*

*lista = [5, 2, 8, 1, 3]*

*#Se crea una nueva lista con los números ordenados, conservando la anterior aún desordenada*

*lista\_ordenada = sorted(lista)*

*print(lista)*

*# Salida: [5, 2, 8, 1, 3]*

*print(lista\_ordenada)*

*# Salida: [1, 2, 3, 5, 8]*

Otro método de ordenamiento que ofrece Python es sort() con un argumento key, que permite especificar una función para determinar la clave de ordenamiento. Por ejemplo, si se tiene una clase[4][5]:

*# Se crea una clase persona con el nombre y la edad como atributos*

*class Persona:*

*def \_\_init\_\_(self, nombre, edad):*

*self.nombre = nombre*

*self.edad = edad*

*def \_\_repr\_\_(self):*

*return f"{self.nombre} ({self.edad} años)"*

*personas = [Persona("Ana", 25), Persona("Juan", 20), Persona("María", 30)]*

Para ordenar la lista de personas por edad, se puede usar sort() con el argumento key[4][5]:

*#Se utiliza el método de ordenamiento con key, especificando que retorne la edad de cada persona*

*personas.sort(key=lambda p: p.edad)*

*print(personas)*

*# Salida: [Juan (20 años), Ana (25 años), María (30 años)]*

sort([key=None][, reverse=False]): Este método ordena los elementos de la lista en orden ascendente. Si se especifica el parámetro reverse=True, se ordenarán en orden descendente. Además, se puede especificar una función key para determinar la clave de ordenamiento. Por ejemplo:

*#Se crea una lista desordenada*

*numeros = [3, 2, 1, 5, 4]*

*# Se utiliza el método de ordenamiento descendente*

*numeros.sort(reverse=True)*

*print(numeros)*

*# Salida: [5, 4, 3, 2, 1]*

*#Se crea una lista de frutas*

*palabras = ['manzana', 'banana', 'naranja', 'pera']*

*#Se utiliza el metodo de ordenamiento para invertir la lista*

*palabras.sort(key=len)*

*print(palabras) # Salida: ['pera', 'banana', 'naranja', 'manzana']*

**2.2 Agregar elementos**

append(elemento): Este método agrega un elemento al final de la lista. Por ejemplo:[3][5]

*frutas = ['manzana', 'banana', 'naranja']*

*frutas.append('pera')*

*print(frutas) # Salida: ['manzana', 'banana', 'naranja', 'pera']*

extend(iterable): Este método agrega todos los elementos de un iterable (por ejemplo, otra lista) al final de la lista original. Por ejemplo[3][5]:

#Se crea una lista de numeros

*numeros = [1, 2, 3]*

*#Se agrega una segunda lista al final de la primera*

*numeros.extend([4, 5, 6])*

*print(numeros) # Salida: [1, 2, 3, 4, 5, 6]*

insert(posicion, elemento): Este método inserta un elemento en la posición especificada de la lista. Por ejemplo[3][5]:

*letras = ['a', 'b', 'd', 'e']*

*letras.insert(2, 'c')*

*print(letras)*

*# Salida: ['a', 'b', 'c', 'd', 'e']*

**2.3 Eliminar elementos**

remove(elemento): Este método elimina la primera aparición de un elemento en la lista. Por ejemplo[3][5]:

*# Se crea una lista de animales*

*animales = ['perro', 'gato', 'ratón']*

*# Se remueve el elemento gato de la lista*

*animales.remove('gato')*

*print(animales)*

*# Salida: ['perro', 'ratón']*

pop([indice]): Este método elimina y retorna el elemento en la posición especificada de la lista. Si no se especifica un índice, se elimina y retorna el último elemento de la lista. Por ejemplo[3][5]:

*# Se crea una lista de números*

*numeros = [1, 2, 3, 4, 5]*

*# Se elimina el último elemento de la lista ya que no se especifica un índice*

*ultimo\_numero = numeros.pop()*

*print(numeros)*

*# Salida: [1, 2, 3, 4]*

*print(ultimo\_numero)*

*# Salida: 5*

**2.4 Otros métodos**

count(elemento): Este método retorna el número de veces que un elemento aparece en la lista.[6] Por ejemplo:

*# Se crea una lista de colores*

*colores = ['rojo', 'verde', 'azul', 'verde', 'rojo', 'verde']*

*# Se pide que retorne la cantidad de veces que se repite el elemento “Verde” en la lista*

*cantidad\_de\_verde = colores.count('verde')*

*print(cantidad\_de\_verde)*

*# Salida: 3*

index(elemento[, inicio[, fin]]): Este método retorna el índice de la primera aparición de un elemento en la lista, dentro del rango especificado. Si el elemento no se encuentra en la lista, retorna un error. Por ejemplo:

*# Se crea una lista de frutas*

*frutas = ["manzana", "plátano", "fresa", "mango", "uva"]*

*# Se busca la posición de la fresa en la lista*

*posicion\_fresa = frutas.index("fresa")*

*# Se imprime la posición de la fresa en la lista*

*print("La fresa está en la posición", posicion\_fresa)*

*# Se intenta buscar una fruta que no está en la lista*

*# Esto generará un error ya que "pera" no está en la lista*

*posicion\_pera = frutas.index("pera")*

**3 Conclusiones**

En Python, las listas son una estructura de datos muy útil y versátil que permite almacenar múltiples elementos en una sola variable. Las listas son mutables, lo que significa que se pueden modificar después de su creación, y pueden contener elementos de diferentes tipos de datos.

Hay muchos métodos útiles disponibles para trabajar con listas en Python, como append(), extend(), insert(), remove(), pop(), index(), count(), sort(), reverse(), entre otros. Cada uno de estos métodos proporciona una funcionalidad específica que puede ser utilizada para manipular y analizar listas de diferentes maneras.

En general, las listas son muy útiles para procesar y analizar datos en Python, especialmente cuando se trabaja con conjuntos de datos complejos y variables. Debido a su flexibilidad y versatilidad, las listas son una herramienta esencial para cualquier programador de Python, y es importante conocer cómo utilizarlas y aplicar sus diferentes métodos para lograr soluciones efectivas y eficientes.

Es importante tener en cuenta que, aunque las listas son muy útiles, también pueden ser propensas a errores si no se utilizan correctamente. Es por eso que es recomendable leer y seguir las guías de estilo de Python y asegurarse de que el código esté limpio y organizado, para evitar errores y problemas en el futuro.

En resumen, las listas son una herramienta fundamental para cualquier programador de Python y ofrecen una amplia variedad de métodos que permiten trabajar con datos de manera efectiva y eficiente. Conocer y utilizar correctamente estos métodos puede ayudar a mejorar la eficacia y eficiencia del código, lo que resulta en una mejor experiencia de programación.

**6.Bibliografía**

[1] PythonSoftwareFundation “FrontPage.”.wiki.python.org <https://wiki.python.org/moin/FrontPage> (Accedido: 24-abr-2023)

[2]A. Ferreira y G.A. Rojo, *Enseñanza de la programación,1ra ed.* Río Cuarto, Argentina. Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología, 2021

[3] ElLibroDePythonTeam “Listas en Python.”ellibrodepython.com.

<https://ellibrodepython.com/listas-en-python> (Accedido: 24-abr-2023)

[4] PythonSoftwareFundation “Sorting HOW TO.” docs.python.org <https://docs.python.org/3/howto/sorting.html#sorting-how-to> (Accedido: 24-abr-2023)

[5] Dimas. Youtube - *¿Cómo funcionan las LISTAS en PYTHON? [con EJERCICIOS RESUELTOS] - Curso PYTHON desde CERO #7.* (20-abr-2021). Accedido: 24-abr-2023. [Video en Línea]. Disponible en: <https://www.youtube.com/watch?v=VsSMC1A7_F0&list=PLVs9JdE6ZZh112EBnYa_wCxYpbVXEXrLF&index=8>

[6] PythonSoftwareFundation “More on List.” docs.python.org <https://docs.python.org/3/tutorial/datastructures.html#more-on-lists> (Accedido: 24-abr-2023)