**1-¿Cuál es la diferencia entre una lista y una tupla en Python?**

Una lista es un tipo de datos mutable en Python que se puede modificar, agregar o eliminar elementos, mientras que una tupla es un tipo de datos inmutable que no se puede modificar después de su creación. Las tuplas se crean con paréntesis (), mientras que las listas se crean con corchetes [].

**2-¿Qué es la recursión? ¿Cuándo es apropiado usarla?**

La recursión es un concepto en programación en el que una función se llama a sí misma para realizar una tarea determinada. Es apropiado usar la recursión cuando la tarea se puede dividir en subproblemas más pequeños y similares al problema original. Ejemplos comunes de problemas que se pueden resolver utilizando la recursión son la búsqueda en profundidad de árboles y la resolución de problemas matemáticos.

**3-¿Qué es la diferencia entre una variable local y una global?**

Una variable local es una variable que se declara dentro de una función y solo existe dentro del ámbito de esa función. Una variable global es una variable que se declara fuera de una función y se puede acceder desde cualquier parte del programa. Las variables globales pueden ser modificadas por cualquier parte del programa, mientras que las variables locales solo pueden ser modificadas dentro de la función en la que se declararon.

**4-¿Qué es un diccionario en Python? ¿Cómo se accede a los elementos de un diccionario?**

Un diccionario es una colección no ordenada de elementos en Python que se almacenan como pares clave-valor. Cada elemento se almacena como una clave única y su valor correspondiente. Los diccionarios se crean con llaves {} y los elementos se separan por comas.

Para acceder a los elementos de un diccionario en Python, se utiliza la clave del elemento en lugar de su índice. Por ejemplo, si tenemos un diccionario llamado poblacion y queremos acceder al valor correspondiente a la clave 'Madrid', podemos hacerlo así: poblacion['Madrid'].

**5-¿Cómo se manejan las excepciones en Python? Proporcione un ejemplo de código que muestre cómo** **manejar una excepción en Python**.

En Python, las excepciones se manejan con un bloque try-except. El código que puede generar una excepción se coloca dentro del bloque try, y el código para manejar la excepción se coloca dentro del bloque except.

Aquí hay un ejemplo de código que maneja una excepción en Python:

pythonCopy code

try:

num = int(input("Ingrese un número entero: ")) resultado = 100 / num print(resultado)

except ZeroDivisionError:

print("No es posible dividir entre cero.")

except ValueError:

print("Debe ingresar un número entero.")

En este ejemplo, se solicita al usuario que ingrese un número entero. Si el usuario ingresa cero, se generará una excepción de ZeroDivisionError cuando se intente dividir 100 entre cero. Si el usuario ingresa algo que no es un número entero, se generará una excepción de ValueError al intentar convertir la entrada en un número entero. Cada excepción se maneja en su propio bloque except, y se imprime un mensaje de error apropiado para cada excepción.

**5. ¿Qué son los decoradores en Python?**

Un decorador es una función que toma otra función y la modifica o mejora su funcionalidad sin cambiar su código original. Los decoradores se escriben como funciones que toman una función como argumento y devuelven otra función que puede ser la misma función modificada o una nueva función con una funcionalidad adicional. Los decoradores son útiles para añadir funcionalidades a una función sin modificar su código original. Por ejemplo, los decoradores se utilizan comúnmente para agregar autenticación, registro o temporización a una función sin tener que modificar el código de la función original.

**6 - ¿Cómo se puede ordenar una lista en Python? Proporcione un ejemplo de código.**

En Python, se puede ordenar una lista utilizando el método sort() para ordenar la lista de forma ascendente o la función sorted() para ordenar la lista de forma ascendente o descendente. Aquí tienes un ejemplo de código:

pythonCopy code

# Ordenar una lista de números de forma ascendente

numbers = [4, 1, 6, 3, 8, 5]

numbers.sort()

print(numbers)

# Ordenar una lista de números de forma descendente

numbers = [4, 1, 6, 3, 8, 5]

numbers\_sorted = sorted(numbers, reverse=True)

print(numbers\_sorted)

# Ordenar una lista de cadenas de texto de forma ascendente

words = ["orange", "banana", "apple", "grape"]

words.sort()

print(words)

**7 - ¿Qué es una función lambda en Python? ¿Cuándo es apropiado usarla?**

Una función lambda en Python es una función anónima que se define en una sola línea de código y no tiene un nombre definido. Las funciones lambda son útiles cuando necesitamos crear una función simple y rápida que se va a utilizar en una única ocasión y no requiere ser definida como una función separada. Las funciones lambda se utilizan comúnmente en combinación con las funciones map(), filter() y reduce() para aplicar operaciones a listas y otras estructuras de datos. Aquí tienes un ejemplo de código:

pythonCopy code

# Definir una función lambda que suma dos números

sum = lambda x, y: x + y

result = sum(3, 5)

print(result)

# Utilizar una función lambda con la función map() para multiplicar cada elemento de una lista por dos

numbers = [1, 2, 3, 4, 5]

doubled\_numbers = list(map(lambda x: x \* 2, numbers))

print(doubled\_numbers)

**8 - ¿Qué es una clase en Python? Proporcione un ejemplo de código que muestre cómo definir una clase en Python.**

Una clase en Python es una estructura de programación que permite definir un tipo de objeto con atributos y métodos. Las clases son utilizadas para crear objetos que pueden ser instanciados y manipulados en el programa. Aquí tienes un ejemplo de código que muestra cómo definir una clase en Python:

pythonCopy code

# Definir una clase de persona class Person: def \_\_init\_\_(self, name, age): self.name = name self.age = age def say\_hello(self): print(f"Hello, my name is {self.name} and I am {self.age} years old.") # Crear un objeto de persona person = Person("John", 30) # Llamar al método say\_hello() del objeto de persona person.say\_hello()

**9 - ¿Cómo se puede leer y escribir archivos en Python? Proporcione un ejemplo de código.**

En Python, se pueden leer y escribir archivos utilizando las funciones open(), read(), write() y close(). Aquí tienes un ejemplo de código que muestra cómo leer y escribir un archivo de texto:

pythonCopy code

# Leer un archivo de texto file = open("file.txt", "r") content = file.read() print(content) file.close() # Escribir en un archivo de texto file = open("file.txt", "w

1-¿Qué es la recursión en Python y cuándo se utiliza?

La recursión es un método en el cual una función se llama a sí misma para resolver un problema. Se utiliza cuando una solución se puede dividir en subproblemas más pequeños que son similares al problema original. Un ejemplo común de uso de la recursión en Python es el cálculo del factorial de un número.

¿Cuál es la diferencia entre una clase y un objeto en Python?

Una clase es una plantilla que define el comportamiento y las propiedades de un objeto, mientras que un objeto es una instancia de una clase. En otras palabras, una clase es como un molde o una plantilla que se utiliza para crear objetos.

¿Cómo se puede crear una función en Python y cuáles son los componentes esenciales de una función?

Para crear una función en Python, se utiliza la sintaxis "def" seguida del nombre de la función y sus parámetros. Los componentes esenciales de una función incluyen su nombre, los parámetros que toma, el cuerpo de la función (donde se define lo que hace la función), y el valor de retorno (si corresponde).

¿Cómo se pueden manejar excepciones específicas en Python?

En Python, se puede manejar excepciones específicas utilizando la sintaxis "try-except". En el bloque "try", se coloca el código que puede generar una excepción, y en el bloque "except", se coloca el código que se ejecutará si se genera una excepción específica.

¿Cómo se pueden comparar dos objetos en Python?

En Python, se puede comparar dos objetos utilizando los operadores de comparación "==" y "!=". Estos operadores comparan el valor de los objetos, mientras que el operador "is" compara las identidades de los objetos.

¿Qué es una lista de comprensión en Python y cómo se utiliza en la práctica?

Una lista de comprensión es una forma concisa y expresiva de crear una nueva lista a partir de otra lista o iterable en Python. Se utiliza para simplificar el código y hacerlo más legible. Por ejemplo, se puede crear una lista de números pares de la siguiente manera: "pares = [x for x in range(10) if x % 2 == 0]".

¿Cómo se pueden importar módulos en Python y qué es un espacio de nombres?

En Python, se pueden importar módulos utilizando la palabra clave "import". Un espacio de nombres es un espacio donde se almacenan los nombres de las variables, funciones y clases. Los módulos en Python tienen su propio espacio de nombres y pueden compartir variables, funciones y clases con otros módulos.

¿Cuál es la diferencia entre un método de instancia y un método de clase en Python?

Un método de instancia es un método que pertenece a un objeto específico, mientras que un método de clase es un método que pertenece a una clase en general. Los métodos de clase se definen utilizando la decoración "@classmethod" y se pueden llamar directamente desde la clase, mientras que los métodos de instancia se definen sin ninguna decoración y se llaman desde un objeto específico.

¿Cómo se puede ordenar una lista en Python y cuáles son los diferentes métodos de ordenamiento disponibles?

En Python, se puede ordenar una lista utilizando el método "sort()" o la función "sorted()". El método "sort()" ordena la lista en su lugar, mientras que la función "sorted()" devuelve una nueva lista ordenada. Algunos de los métodos de ordenamiento disponibles en Python incluyen la ordenación por burbuja, la ordenación por selección, la ordenación por inserción, la ordenación por mezcla y la ordenación rápida.

¿Cómo se pueden manejar los errores de tiempo de ejecución en Python y cuáles son los errores comunes que se pueden encontrar?

En Python, se pueden manejar los errores de tiempo de ejecución utilizando la sintaxis "try-except". Algunos de los errores comunes de tiempo de ejecución que se pueden encontrar en Python incluyen la división por cero, el índice fuera de rango, el tipo de dato incorrecto y los errores de archivo. Es importante manejar estos errores adecuadamente para garantizar que el programa se ejecute correctamente y que los usuarios obtengan una experiencia sin errores.

1-¿Cómo se pueden manejar archivos en Python y cuál es la forma correcta de cerrar un archivo después de su uso?

En Python, se puede manejar archivos utilizando la función "open()" para abrir un archivo y la función "close()" para cerrar el archivo después de su uso. Es importante cerrar el archivo después de su uso para liberar los recursos del sistema operativo. Además, también se puede utilizar la estructura "with" para manejar archivos en Python, lo que asegura que el archivo se cierre automáticamente después de su uso. Por ejemplo:

python

Copy code

# Abrir un archivo en modo de lectura

archivo = open("archivo.txt", "r")

# Leer el contenido del archivo

contenido = archivo.read()

# Cerrar el archivo

archivo.close()

# Manejo de archivos con la estructura "with"

with open("archivo.txt", "r") as archivo:

contenido = archivo.read()

**2-¿Cómo se puede iterar sobre los elementos de un diccionario en Python?**

En Python, se puede iterar sobre los elementos de un diccionario utilizando un bucle "for". Por defecto, el bucle "for" itera sobre las claves del diccionario, pero se puede acceder a los valores del diccionario utilizando la sintaxis de indexación. Además, también se puede utilizar el método "items()" del diccionario para iterar sobre los pares clave-valor del diccionario. Por ejemplo:

python

Copy code

mi\_diccionario = {"nombre": "Juan", "edad": 30, "ciudad": "Bogotá"}

# Iterar sobre las claves del diccionario

for clave in mi\_diccionario:

print(clave)

# Iterar sobre los valores del diccionario

for clave in mi\_diccionario:

print(mi\_diccionario[clave])

# Iterar sobre los pares clave-valor del diccionario

for clave, valor in mi\_diccionario.items():

print(clave, valor)

**3-¿Cómo se puede crear una clase en Python y cuáles son los componentes esenciales de una clase?**

En Python, se puede crear una clase utilizando la palabra clave "class", seguida del nombre de la clase y dos puntos. Dentro de la clase, se pueden definir atributos y métodos que pertenecen a la clase. Los componentes esenciales de una clase son:

El constructor de la clase, que se llama "init()" y se utiliza para inicializar los atributos de la clase.

Los atributos de la clase, que son variables que pertenecen a la clase y se definen dentro del constructor o directamente en la clase.

Los métodos de la clase, que son funciones que pertenecen a la clase y se definen dentro de la clase. Los métodos pueden acceder y modificar los atributos de la clase. Por ejemplo:

python

Copy code

class Persona:

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

self.nombre = nombre

self.edad = edad

def saludar(self):

print("Hola, mi nombre es", self.nombre, "y tengo", self.edad, "años.")

juan = Persona("Juan", 30)

juan.saludar()