Fundamentos generales de la programación con el Lenguaje Python

Actividad No. 9: Realizar la lectura de este material y elaborar una **INFOGRAFÍA** digital completa con multimedia, referencias, audio o podcasts, imágenes, textos, ejemplos, conceptos, concisos sobre el siguiente contenido:

La siguiente clase se evaluara esta actividad para todos los estudiantes.

¿Qué significa el logo que utiliza Python?

El logo de Python es una imagen de una serpiente pitón enroscada en sí misma. Fue creado en 1991 por Guido van Rossum, el creador de Python, quien eligió una serpiente como imagen porque es un animal que puede ser encontrado en todo el mundo y es utilizado en muchas culturas como símbolo de sabiduría.

El diseño del logo de Python fue creado por el diseñador gráfico holandés, Just van Rossum (sin relación con Guido van Rossum). La imagen representa una serpiente pitón que se enrosca en sí misma, lo que simboliza la naturaleza flexible y escalable de Python. Además, la elección de una serpiente pitón es una referencia directa al nombre del lenguaje, ya que la serpiente pitón es una especie de serpiente grande y poderosa, lo que refleja la capacidad de Python para manejar proyectos de gran escala.

En resumen, el logo de Python representa la **flexibilidad, escalabilidad** y **potencia** del lenguaje, y la elección de una serpiente pitón como imagen es una referencia directa al nombre del lenguaje y su capacidad para manejar proyectos de gran escala.

¿Qué es el lenguaje de programación Python?

Python es un lenguaje de programación de alto nivel, interpretado y generalmente utilizado para el desarrollo de software, aplicaciones web y análisis de datos. Fue **creado en 1989 por Guido van Rossum y** se ha convertido en uno de los lenguajes

de programación más populares en la actualidad debido a su simplicidad, legibilidad y facilidad de uso.

Las características de Python, podemos mencionar:

Es un lenguaje interpretado, lo que significa que no necesita ser compilado antes de su ejecución.

Es multiplataforma y puede ser utilizado en diversos sistemas operativos, como Windows, Linux y MacOS.

Posee una sintaxis clara y sencilla, lo que facilita la lectura y escritura de código.

Es un lenguaje orientado a objetos, lo que permite la encapsulación, herencia y polimorfismo. Complementar esta información y registrarla en su multimedia.

Es un lenguaje con tipado dinámico, lo que significa que las variables no necesitan ser definidas antes de su uso.

Python se utiliza en una amplia variedad de campos, como el **desarrollo web**, **la inteligencia artificial, el análisis de datos**, la **automatización de tareas**, **la creación de videojuegos y la ciberseguridad.**

¿Cuáles son las bibliotecas de Python más importantes y para que se utiliza cada una?

Python tiene una **amplia gama de bibliotecas**, pero algunas de las más importantes y utilizadas son las siguientes:

NumPy: es una biblioteca para el procesamiento de datos numéricos en Python. Proporciona una variedad de herramientas para realizar cálculos matemáticos y científicos, y se utiliza comúnmente en la ciencia de datos y el aprendizaje automático.

Pandas: es una biblioteca para el análisis de datos en Python. Proporciona estructuras de datos de alto rendimiento y herramientas para manipular y analizar datos, y se utiliza ampliamente en la ciencia de datos y el análisis financiero.

Matplotlib: es una biblioteca para la visualización de datos en Python. Permite la creación de gráficos y visualizaciones en 2D y 3D, y se utiliza en la ciencia de datos, la ingeniería y otras disciplinas para la visualización de datos.

Scikit-learn: es una biblioteca para el aprendizaje automático en Python. Proporciona una variedad de herramientas para la clasificación, regresión, agrupamiento y preprocesamiento de datos, y se utiliza en la ciencia de datos, la ingeniería y otras disciplinas para la construcción de modelos de aprendizaje automático.

TensorFlow: es una biblioteca para el aprendizaje automático en Python. Proporciona una variedad de herramientas para la construcción de redes

neuronales y modelos de aprendizaje profundo, y se utiliza en la ciencia de datos, la ingeniería y otras disciplinas para el aprendizaje profundo y el procesamiento del lenguaje natural.

Django: es un marco de trabajo para el desarrollo de aplicaciones web en Python. Proporciona herramientas para la creación de aplicaciones web de alta calidad y escalables, y se utiliza ampliamente en la industria para el desarrollo de sitios web y aplicaciones web.

Estas son solo algunas de las bibliotecas más importantes en Python, y cada una se utiliza para un propósito específico en la **ciencia de datos**, la **ingeniería** y otras disciplinas.

En resumen, estas son solo algunas de las bibliotecas más utilizadas en Python son NumPy para el procesamiento numérico, Pandas para el análisis de datos y Django para el desarrollo web. Además, empresas como Google, Facebook y Netflix utilizan Python en sus aplicaciones y servicios.

¿Cuál es la sintaxis que utiliza Python como lenguaje de programación?

La sintaxis de Python es conocida por ser legible y fácil de entender debido a que utiliza una sintaxis clara y sencilla. A continuación se muestran algunas características de la sintaxis de Python:

Indentación: Python utiliza la indentación para marcar los bloques de código, en lugar de llaves como en otros lenguajes. Esto significa que la indentación es obligatoria en Python y se utiliza para definir la estructura del código.

Tipos de datos: Python tiene varios tipos de datos incorporados, como enteros, flotantes, cadenas y booleanos. También hay estructuras de datos incorporadas como listas, tuplas y diccionarios.

Variables: En Python, las variables no necesitan ser declaradas explícitamente antes de ser utilizadas. Simplemente se les asigna un valor y Python determina su tipo de dato automáticamente.

Funciones: En Python, las funciones se definen con la palabra clave "def" y pueden tomar argumentos opcionales. Las funciones pueden devolver valores o no devolver ninguno.

Control de flujo: Python utiliza estructuras de control de flujo como "if", "else" y "while" para controlar el flujo del programa.

Ejemplo explicativo de la sintaxis de Python:

```
# Definir una función que suma dos números

def suma(a, b):

    resultado = a + b
```

```
    return resultado


# Llamar a la función y mostrar el resultado
x = 3
y = 5
z = suma(x, y)
print(z) # Mostrará "8" en la consola
```

En resumen, la sintaxis de Python se caracteriza por su legibilidad, su uso de la indentación para marcar los bloques de código y su gran cantidad de tipos de datos y estructuras de control de flujo incorporadas.

¿Qué es una variable en Python?

En Python, una variable es un nombre que se utiliza para hacer referencia a un valor que se almacena en la memoria del computador. Las variables se utilizan para almacenar valores y hacer referencia a ellos en el código.

Las variables en Python son dinámicamente tipadas, lo que significa que no es necesario declarar el tipo de dato de una variable antes de utilizarla. El tipo de dato se determina automáticamente en tiempo de ejecución según el valor asignado a la variable.

Para crear una variable en Python, simplemente se le asigna un valor utilizando el operador de asignación "=" seguido del valor. Por ejemplo, la siguiente línea de código crea una variable llamada "nombre" y le asigna el valor "Juan":

```
nombre = "Juan"
```

En este caso, la variable "nombre" es una cadena de caracteres, pero Python no requiere que se especifique su tipo de dato de antemano.

Las variables pueden ser utilizadas en diferentes partes del código y su valor puede cambiar durante la ejecución del programa. Por ejemplo, se puede cambiar el valor de la variable "nombre" como se muestra a continuación:

```
nombre = "Juan"
nombre = "María"
```

En este caso, la variable "nombre" se redefine con un valor diferente después de la primera asignación.

En resumen, una variable en Python es un nombre utilizado para hacer referencia a un valor almacenado en la memoria de la computadora. Las variables en Python

son dinámicamente tipadas y su valor puede cambiar durante la ejecución del programa.

¿Qué es una función en Python?

En Python, una función es un bloque de código reutilizable que se puede llamar desde otras partes del programa. Las funciones son una forma de modularizar el código y separar la funcionalidad en bloques lógicos y fáciles de entender. Las funciones también permiten reutilizar el código sin tener que escribirlo varias veces en diferentes partes del programa.

Una función en Python se define utilizando la palabra clave "def", seguida del nombre de la función y los parámetros entre paréntesis. Dentro de la función, se puede escribir el código que se ejecutará cuando se llame a la función. La función puede tener un valor de retorno opcional utilizando la palabra clave "return".

Ejemplo simple de una función en Python:

```
def suma(a, b):
    resultado = a + b
    return resultado
```

En este caso, la función "suma" toma dos argumentos "a" y "b" y devuelve su suma. La función se puede llamar desde otras partes del programa, pasando los argumentos deseados. Por ejemplo:

```
x = 3
y = 5
z = suma(x, y)
print(z) # Mostrará "8" en la consola
```

En este ejemplo, la función "suma" se llama con los argumentos "x" e "y" y devuelve su suma en la variable "z". Luego, el valor de "z" se muestra en la consola utilizando la función "print".

En resumen, una función en Python es un bloque de código reutilizable que se puede llamar desde otras partes del programa. Las funciones se definen utilizando la palabra clave "def" y pueden tener argumentos y valores de retorno opcionales. Las funciones son una herramienta poderosa para modularizar y reutilizar el código en Python.

¿Qué es una condicional en Python y cómo se representa?

En Python, una condicional es una estructura de control que permite ejecutar diferentes bloques de código dependiendo de una condición dada. En otras

palabras, una condicional permite que el programa tome una decisión y ejecute diferentes acciones en función del resultado de una evaluación lógica.

La sintaxis de una condicional en Python se realiza con la palabra clave "if" seguida de una expresión booleana que se evalúa como Verdadero o Falso. Si la expresión booleana es Verdadero, se ejecuta el bloque de código que sigue al "if". Si la expresión es Falso, se puede proporcionar un bloque alternativo de código que se ejecutará utilizando la palabra clave "else".

Aquí hay un ejemplo sencillo de una condicional en Python:

```
x = 10

if x > 0:

    print("x es positivo")

else:

    print("x es negativo o cero")
```

En este caso, la condicional comprueba si el valor de "x" es mayor que cero. Si es así, se imprimirá "x es positivo" en la consola. Si no, se imprimirá "x es negativo o cero". En este ejemplo, la expresión booleana se evalúa como Verdadero y se ejecuta el bloque de código que sigue al "if".

Además del "if" y "else", también es posible utilizar múltiples expresiones "elif" para crear condicionales más complejas que evalúan varias opciones. La sintaxis de una condicional con "elif" se vería así:

```
x = 10

if x > 0:

    print("x es positivo")

elif x == 0:

    print("x es cero")

else:

    print("x es negativo")
```

En este caso, si el valor de "x" es mayor que cero, se imprimirá "x es positivo". Si es igual a cero, se imprimirá "x es cero". Si es menor que cero, se imprimirá "x es negativo". En este ejemplo, la expresión booleana se evalúa como Verdadero y se ejecuta el bloque de código que sigue al "if".

En resumen, una condicional en Python es una estructura de control que permite ejecutar diferentes bloques de código dependiendo de una condición dada. La sintaxis se realiza con las palabras clave "if", "else" y "elif" y se evalúa una expresión booleana para tomar una decisión. Las condicionales son una herramienta importante en la programación para hacer que el código sea más flexible y adaptable a diferentes situaciones.

¿Qué es un ciclo o bucle en python y como se usan?

En Python, un ciclo o bucle es una estructura de control que permite repetir un bloque de código varias veces, hasta que se cumpla una condición específica. Los bucles son una herramienta importante en la programación para automatizar tareas repetitivas y procesar grandes cantidades de datos.

Hay dos tipos de bucles en Python: el bucle "while" y el bucle "for". El bucle "while" se utiliza para repetir un bloque de código mientras se cumpla una condición, mientras que el bucle "for" se utiliza para iterar sobre una secuencia de valores, como una lista o una cadena.

Aquí hay un ejemplo de un bucle "while" en Python:

```
x = 0

while x < 5:
    print(x)
    x += 1
```

En este caso, el bucle "while" se repetirá mientras la variable "x" sea menor que 5. En cada iteración del bucle, se imprimirá el valor de "x" en la consola y se aumentará en 1. El bucle se detendrá cuando el valor de "x" sea igual o mayor que 5.

Ejemplo de un bucle "for" en Python:

```
frutas = ["manzana", "naranja", "platano"]

for fruta in frutas:
    print(fruta)
```

En este caso, el bucle "for" se utiliza para iterar sobre una lista de frutas. En cada iteración del bucle, se asigna el valor de la fruta actual a la variable "fruta" y se imprime en la consola. El bucle se detendrá después de que se hayan procesado todos los elementos de la lista.

Los bucles son una herramienta importante en la programación para automatizar tareas repetitivas y procesar grandes cantidades de datos. En Python, se pueden utilizar los bucles "while" y "for" para iterar sobre secuencias de valores y ejecutar un bloque de código varias veces, hasta que se cumpla una condición específica.

ue son los operadores y cuales son en Python?

En programación, los operadores son símbolos o palabras reservadas que realizan operaciones matemáticas o lógicas en uno o varios valores. En Python, existen varios tipos de operadores, que se clasifican en diferentes categorías según su función y los tipos de datos que manipulan. A continuación, se describen los principales operadores en Python:

Operadores aritméticos: se utilizan para realizar operaciones matemáticas entre dos o más valores. Los operadores aritméticos en Python son: "+" (suma), "-" (resta), "*" (multiplicación), "/" (división), "%" (módulo) y "**" (potencia).

Operadores de comparación: se utilizan para comparar dos valores y producir un valor booleano (verdadero o falso) como resultado. Los operadores de comparación en Python son: "==" (igual a), "!=" (distinto de), ">" (mayor que), ">=" (mayor o igual que), "<" (menor que) y "<=" (menor o igual que).

Operadores lógicos: se utilizan para combinar dos o más valores booleanos y producir un valor booleano como resultado. Los operadores lógicos en Python son: "and" (y), "or" (o) y "not" (no).

Operadores de asignación: se utilizan para asignar un valor a una variable. El operador de asignación básico en Python es el signo "=".

Operadores de identidad: se utilizan para comparar la identidad de dos objetos. Los operadores de identidad en Python son: "is" (es) y "is not" (no es).

Operadores de pertenencia: se utilizan para comprobar si un valor pertenece a una secuencia. Los operadores de pertenencia en Python son: "in" (en) y "not in" (no en).

En general, los operadores en Python se utilizan para realizar operaciones matemáticas y lógicas en diferentes tipos de datos, como números, cadenas, listas, tuplas, conjuntos, entre otros.

Actividad No. 10: Verificación programando en Python de cada uno de los ejemplos que presenta el material sobre fundamentos de este lenguaje de programación fundamentados anteriormente.

Actividad No. 10 segunda parte:

Prácticas resueltas utilizando el lenguaje programando en Python: Verificación y trabajo en clase:

Práctica resuelta No. 1*:* Definir una **función** llamada ***num_max()*** la cual nos va a mostrar por pantalla cuál de tres números enteros digitados (m,n,r) es el mayor y un aviso que nos informa que los números ingresados son iguales. El código de programación que responde a la solución del problema es el siguiente:

```
m = int(input("Ingrese el primer número entero: "))
n = int(input("Ingrese el segundo número entero: "))
r = int(input("Ingrese el tercer número entero: "))
if m > n and m > r:
    print("El número mayor es:", m)
elif n > m and n > r:
    print("El número mayor es:", n)
elif r > m and r > n:
    print("El número mayor es:", r)
else:
    print("Los números ingresados son iguales")
```

Análisis y explicación del código:

Este código utiliza la función input() para que el usuario registre el ingreso de los tres números enteros por teclado. Los números solictados corresponden a m, n y r.

Se hace uso de una estructura condicional if-elif-else.

Posteriormente se utiliza una estructura condicional if-elif-else para comparar los números ingresados (m, n, r) y así determinar cuál es el mayor de ellos. Si los números ingresados son iguales, se imprime un mensaje indicando que son iguales.

Practica resuelta No. 2: Definir una **función** llamada ***num_max_min()*** la cual nos va a mostrar por pantalla cuál de tres números enteros digitados (m,n,r) es el mayor y cuál es el número entero menor y un aviso que nos informa que los números ingresados son iguales.

```
def num_max_min(m, n, r):
    if m > n and m > r:
        mayor = m
        if n > r:
            menor = r
        else:
            menor = n
    elif n > m and n > r:
        mayor = n
        if m > r:
            menor = r
```

Análisis y explicación del código:

Este ejercicio nos lleva a definir la función num_max_min que permite establecer entre los números m.n y r, cuál de ellos es el mayor y y cuál es el menor. La función en cuestión, recibe los tres números m, n y r como parámetros, y devuelve una cadena que indica cuál es el número mayor y cuál es el número menor.

Lo que se ha hecho, es definir una función num_max_min que recibe tres números enteros m, n y r. Dentro de esta función, empleamos una estructura condicional if-elif-else para determinar cuál es el número mayor entre ellos. Luego, se anida otra estructura condicional dentro de cada rama if-elif para determinar cuál

`` else: menor = m elif r > m and r > n: mayor = r if m > n: menor = n else: menor = m else: return "Los números ingresados son iguales" return f"El mayor es {mayor} y el menor es {menor}" # Pedimos al usuario que ingrese los tres números m = int(input("Ingrese el primer número entero: ")) n = int(input("Ingrese el segundo número entero: ")) r = int(input("Ingrese el tercer número entero: ")) # Llamamos a la función num_max_min y mostramos el resultado resultado = num_max_min(m, n, r) print(resultado) ```	es el número menor entre los dos números restantes. Finalmente, si los tres números son iguales, la función devuelve una cadena que indica que los números son iguales. Si no, la función devuelve una cadena que indica cuál es el número mayor y cuál es el número menor. Fuera de la función, se requiere al usuario para que ingrese los tres números enteros usando la función input(), y luego se llama a la función num_max_min pasándole los tres números como argumentos. El resultado de la función se almacena en la variable resultado, que luego se muestra en la pantalla usando la función print(). Además se muestra un aviso informando que los números ingresados por el usuario son iguales.

Práctica resuelta No. 3: Dibujar un cuadrado de color verde y grosor de pluma 5 y en la esquina inferior izquierda del cuadrado, dibujar haciendo coincidir el vértice con el perímetro de una circunferencia de color azul con grosor de pluma 5.

```
import turtle


# 2. Crear una ventana en blanco para dibujar la figura
ventana = turtle.Screen()
ventana.bgcolor("white")
# 3. Crear un objeto turtle
tortuga = turtle.Turtle()
# 4. Configurar el tamaño de la pluma y el color
tortuga.pensize(5)
tortuga.pencolor("blue")
# 5. Dibujar el radio en color rojo
tortuga.pencolor("red")
tortuga.forward(50)
tortuga.penup()
tortuga.backward(50)
tortuga.pendown()


# 6. Dibujar el diámetro en color verde
tortuga.pencolor("green")
tortuga.left(90)
tortuga.forward(100)
tortuga.right(90)
tortuga.forward(100)
tortuga.right(90)
tortuga.forward(100)
tortuga.right(90)
tortuga.forward(100)
```

Para dibujar una circunferencia en Python utilizando la biblioteca Turtle, puedes seguir estos pasos:

1. Importar la biblioteca turtle

2. Crear una ventana en blanco para dibujar la figura

3. Crear un objeto turtle

4. Configurar el tamaño de la pluma y el color

5. Dibujar el radio en color rojo.

5. Dibujar el diámetro en color verde.

6. Dibujar la circunferencia en color azul.

7. Cerrar la ventana.

La función turtle.Screen() crea una ventana en blanco con un fondo blanco y la asigna a la variable ventana.

La función turtle.Turtle() crea un objeto turtle que se llama tortuga. Este objeto se puede mover para dibujar en la ventana.

Las funciones tortuga.pensize(5) y tortuga.pencolor("blue") establecen el tamaño de la pluma en 5 y el color de la pluma en azul, respectivamente.

Las funciones tortuga.forward(50), tortuga.penup(), tortuga.backward(50), y tortuga.pendown() dibujan el radio de la circunferencia en color rojo.

Las funciones tortuga.left(90), tortuga.forward(100), tortuga.right(90), y tortuga.forward(100) se repiten tres veces para dibujar el cuadrado del diámetro en color verde.

La función tortuga.penup() levanta la pluma del papel, la función tortuga.goto(0, 0) mueve la tortuga al centro de la ventana, y la función

# 7. Dibujar la circunferencia en color azul tortuga.pencolor("blue") tortuga.penup() tortuga.goto(0, 0) tortuga.pendown() tortuga.circle(50) # 8. Cerrar la ventana ventana.mainloop()	tortuga.pendown() baja la pluma del papel para continuar dibujando. La función tortuga.circle(50) dibuja la circunferencia en color azul. Finalmente, la función ventana.mainloop() inicia el ciclo principal de la ventana para que permanezca abierta hasta que el usuario la cierre.

Practica resuelta No. 4: Dibujar un octágono en color rojo y dentro de él una circunferencia de color azul. La pluma debe tener grosor de 4.

import turtle # Crear objeto turtle t = turtle.Turtle() # Definir medida del octágono y de la circunferencia radio = 100 lado = radio * 2 ** 0.5 apotema = radio * (2 - 2 ** 0.5) # Dibujar el octágono t.color("red", "yellow") t.begin_fill() t.pensize(5)	Análisis y explicación del código: El código utiliza el módulo turtle para dibujar un octágono y una circunferencia inscrita en él. A continuación se explica línea por línea: import turtle Se importa el módulo turtle. t = turtle.Turtle() Se crea un objeto turtle llamado "t". radio = 100 lado = radio * 2 ** 0.5 apotema = radio * (2 - 2 ** 0.5) Se definen las medidas del octágono y de la circunferencia inscrita en función del radio. t.color("red", "yellow")

t.penup() t.goto(-lado/2, -apotema) t.pendown() for i in range(8): t.forward(lado) t.left(45) t.end_fill() # Dibujar la circunferencia inscrita t.color("blue", "green") t.begin_fill() t.penup() t.goto(0, 0) t.pendown() t.circle(radio) t.end_fill() # Mantener la ventana abierta turtle.done()	t.begin_fill() t.pensize(5) t.penup() t.goto(-lado/2, -apotema) t.pendown() for i in range(8): t.forward(lado) t.left(45) t.end_fill() Se dibuja el octágono relleno de amarillo y bordeado de rojo. Para ello se define el color de la pluma y del relleno, se inicia el relleno con begin_fill(), se configura el tamaño de la pluma con pensize(), se levanta la pluma con penup(), se mueve la tortuga a la posición inicial con goto(), se baja la pluma con pendown(), y se utiliza un bucle for para dibujar los lados del octágono, avanzando la distancia lado y girando 45 grados a la izquierda en cada iteración. Finalmente se cierra el relleno con end_fill(). t.color("blue", "green") t.begin_fill() t.penup() t.goto(0, 0) t.pendown() t.circle(radio) t.end_fill() Se dibuja la circunferencia inscrita en el octágono, rellena de verde y bordeada de azul. Para ello se define el color de la pluma y del relleno, se inicia el relleno con begin_fill(), se levanta la pluma con penup(), se mueve la

	tortuga al centro del octágono con goto(), se baja la pluma con pendown(), y se utiliza la función circle() para dibujar la circunferencia inscrita, utilizando el valor del radio previamente definido. Finalmente se cierra el relleno con end_fill(). turtle.done() Mantiene la ventana abierta hasta que se cierra manualmente.

Practica resuelta No. 5 Dibujar un cuadrado y cada uno de sus lados con un color diferente azul, violeta, verde y rojo y obtener su área y su perímetro, conociendo que el lado mide 300 unidades lineales.

import turtle # Configurar la ventana de dibujo ventana = turtle.Screen() ventana.bgcolor("Black") # Configurar la tortuga para dibujar el cuadrado tortuga = turtle.Turtle() tortuga.pensize(6) # Mover la tortuga abajo de la ventana de dibujo tortuga.penup() tortuga.goto(-100, -100) tortuga.pendown() # Dibujar el cuadrado lado = 300	La primera línea importa el módulo "turtle". La segunda línea crea una ventana de dibujo y la asigna a la variable "ventana", y configura su color de fondo en negro. La tercera línea crea un objeto "tortuga" que se utilizará para dibujar y lo asigna a la variable "tortuga", y configura el tamaño de la pluma en 6. Las siguientes tres líneas mueven la tortuga hacia abajo de la ventana de dibujo y la colocan en la esquina inferior izquierda del cuadrado que se dibujará. La línea "lado = 300" define el tamaño de cada lado del cuadrado. Las siguientes cuatro líneas dibujan el cuadrado utilizando la tortuga. La tortuga se mueve hacia adelante el tamaño del lado y gira 90 grados a la izquierda antes de repetir el proceso tres veces más. Las dos líneas siguientes calculan el área y el perímetro del cuadrado. Las dos últimas líneas imprimen el área y el perímetro en la consola.

tortuga.pencolor("blue") tortuga.forward(lado) tortuga.pencolor("violet") tortuga.left(90) tortuga.forward(lado) tortuga.pencolor("green") tortuga.left(90) tortuga.forward(lado) tortuga.pencolor("red") tortuga.left(90) tortuga.forward(lado) # Calcular el área y el perímetro area = lado ** 2 perimetro = lado * 4 # Mostrar el área y el perímetro en la consola print("El área del cuadrado es:", area) print("El perímetro del cuadrado es:", perimetro) # Esperar a que el usuario cierre la ventana de dibujo turtle.done()	La última línea espera a que el usuario cierre la ventana de dibujo antes de terminar la ejecución del programa.