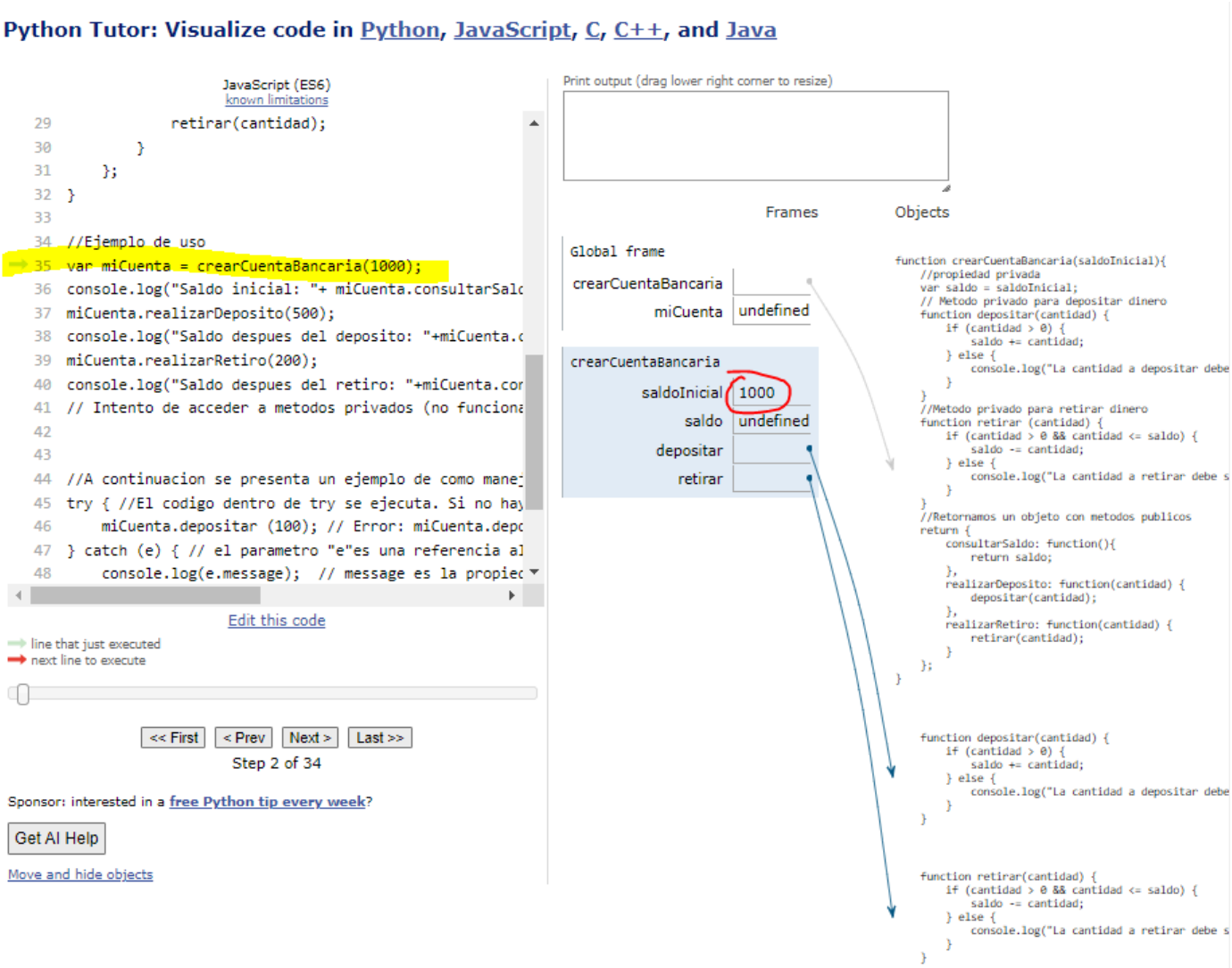
- Crea un repositorio en GitHub donde subirás tu trabajo. Asegúrate de que el repositorio sea público para que podamos revisarlo.
- Dentro del repositorio, crea un archivo para el programa que se presenta a continuación. El archivo debe tener extensión .js.
- Copia el programa en Pythontutor.com y ejecuta el programa, línea por línea, realiza una captura de cada ejecución en donde se observe el resultado de cada ejecución, son 34 pasos, deberás explicar de manera breve y sencilla cada uno de los pasos. Crea un archivo PDF que contenga todas las capturas. Este archivo debe estar bien estructurado y presentado de manera clara.

1. Comenzaremos detallando el codigo de la funcion fabrica **crearCuentaBancaria(saldoInicial)**:

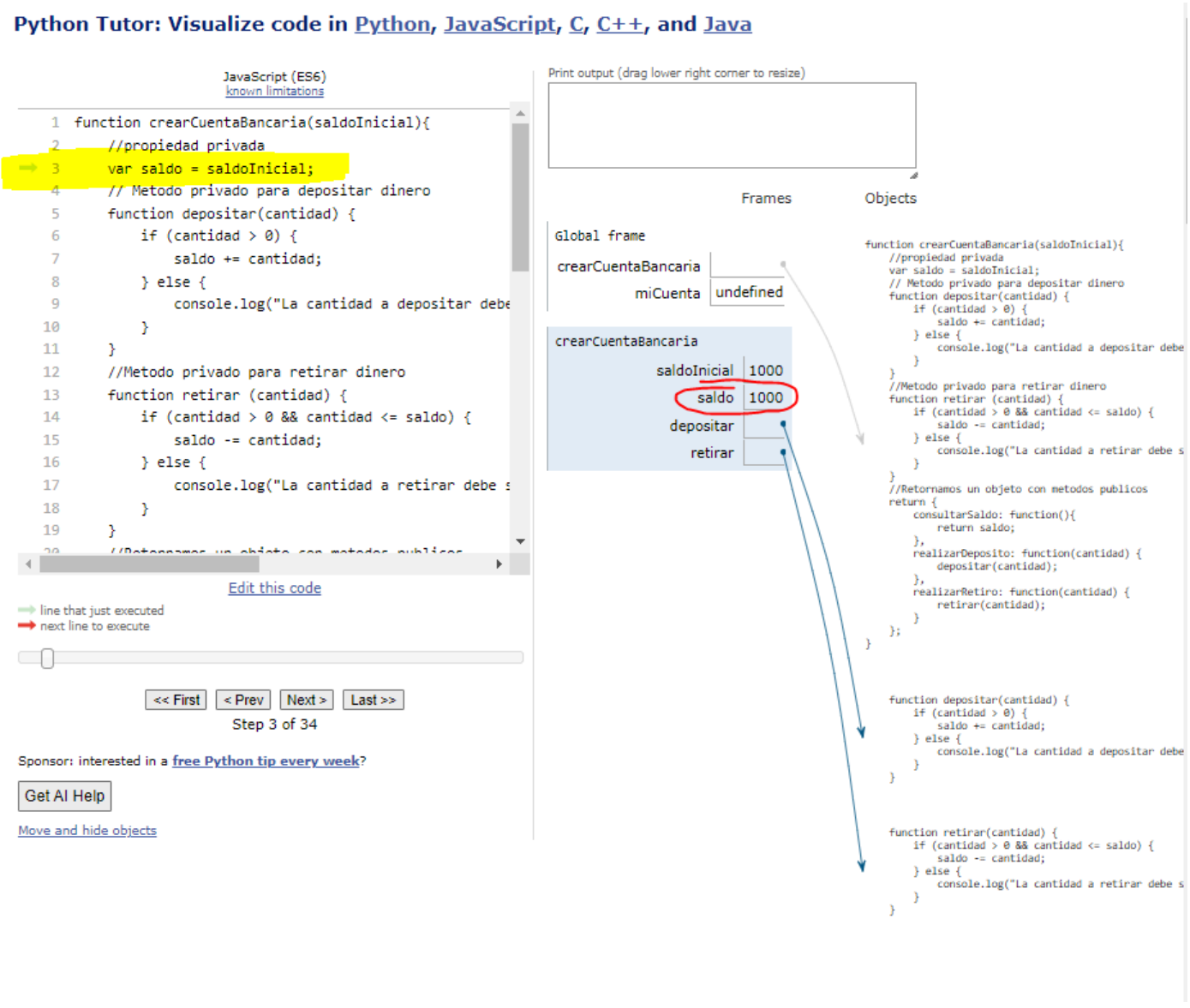
```
// Funcion fabrica para crear una cuenta bancaria
function crearCuentaBancaria(saldoInicial){
    //propiedad privada
    var saldo = saldoInicial;
    // Metodo privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero");
        }
    }
    //Metodo privado para retirar dinero
    function retirar (cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no exceder
el saldo disponible");
        }
    }
    //Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function(){
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

2. **`var miCuenta = crearCuentaBancaria(1000);`**
   Declaramos variable var "miCuenta" que contendrá la función "crearCuentaBancaria" con un saldo inicial de "1000".

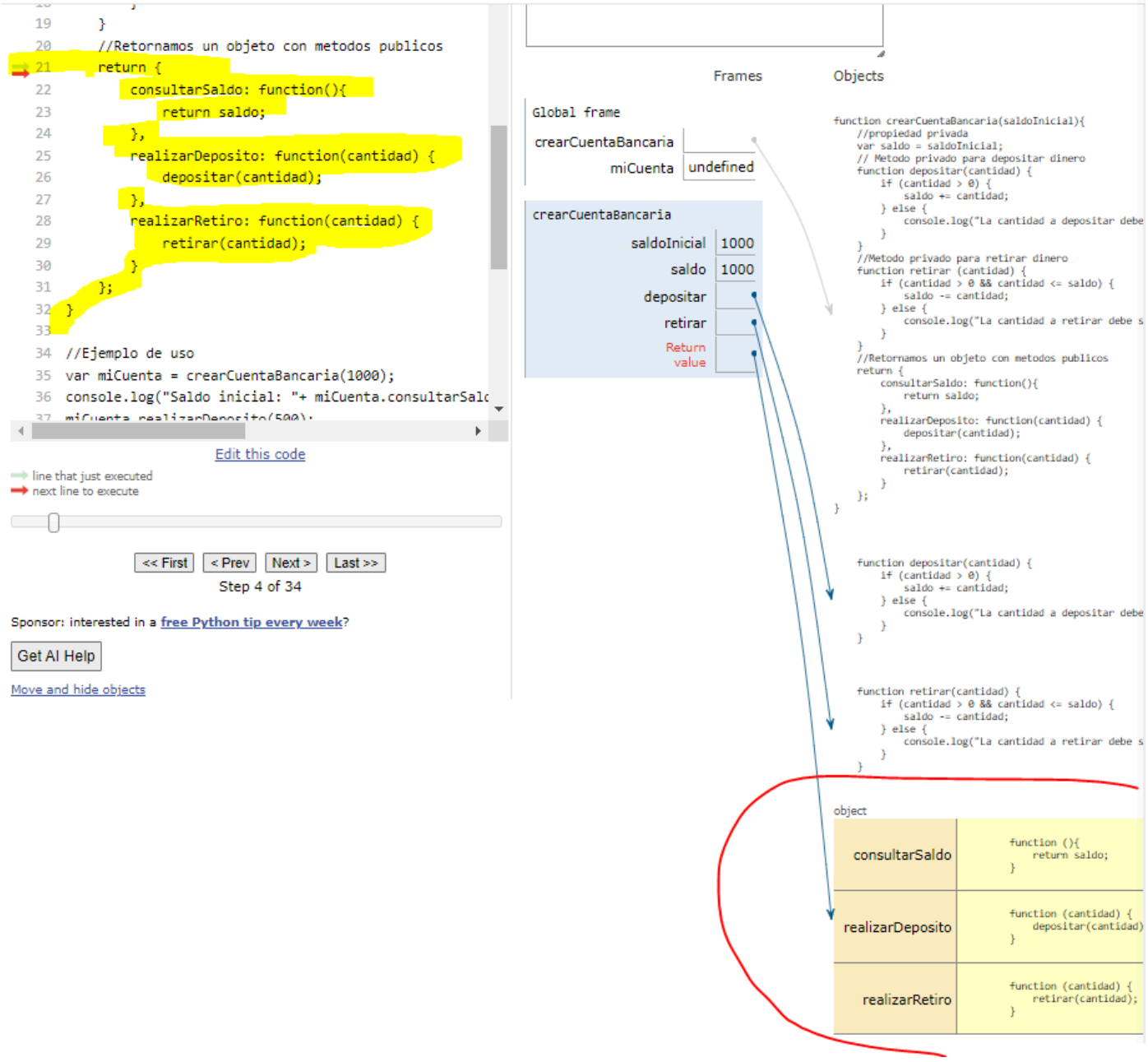


3. **`var saldo = saldoInicial;`**

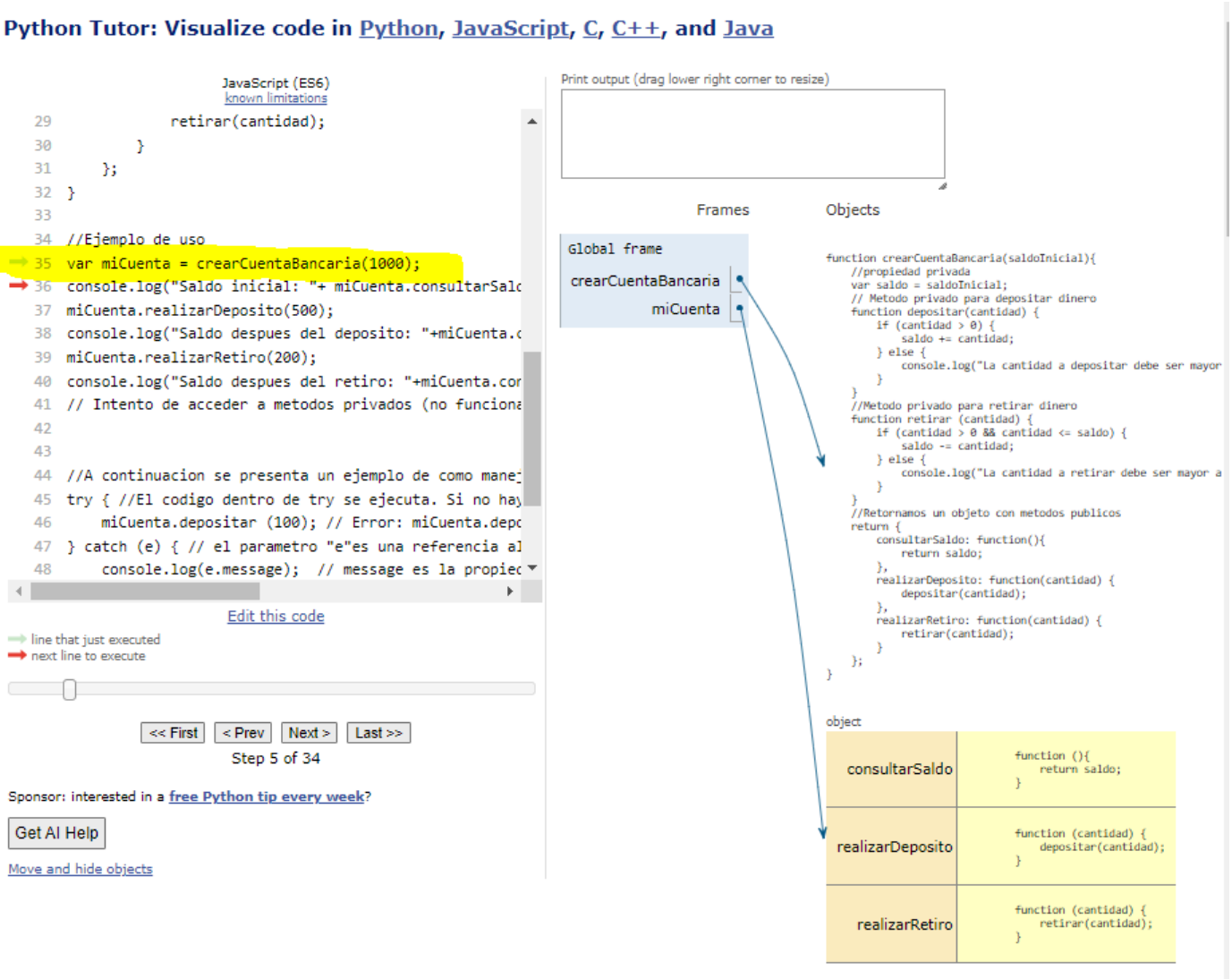
   Declaracion de propiedad privada var “saldo” dentro de la funcion “crearCuentaBancaria(saldoInicial)”. Esta contiene el valor de “saldoInicial” declarado en la funcion, que en este caso es 1000.



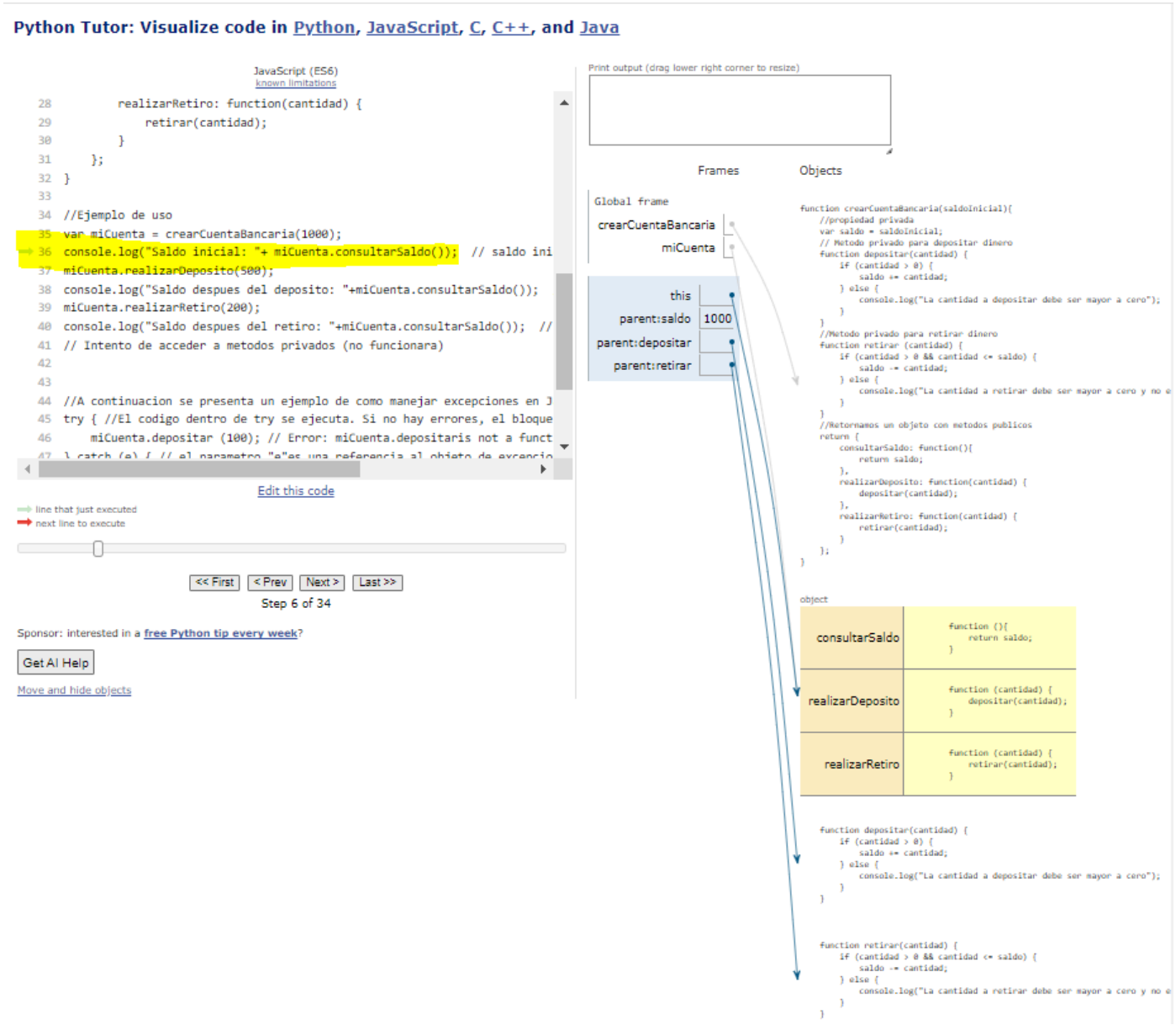
4. La función se ha ejecutado y ahora retornará el resultado objeto obtenido después de haber recorrido la función "crearCuentaBancaria".



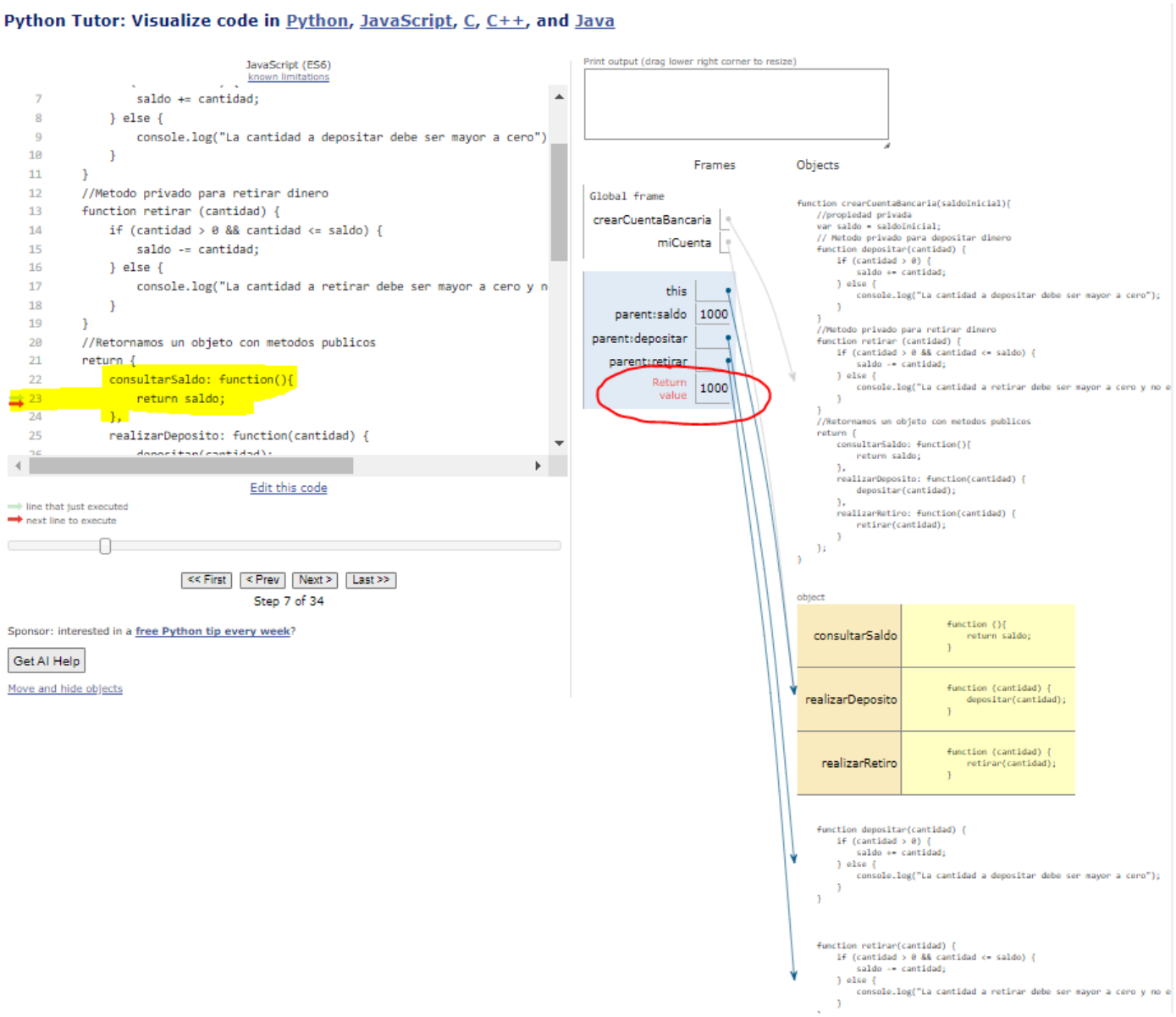
5. **`var miCuenta = crearCuentaBancaria(1000);`** ya se ha terminado de ejecutar despues de haber recorrido la funcion "crearCuentaBancaria"



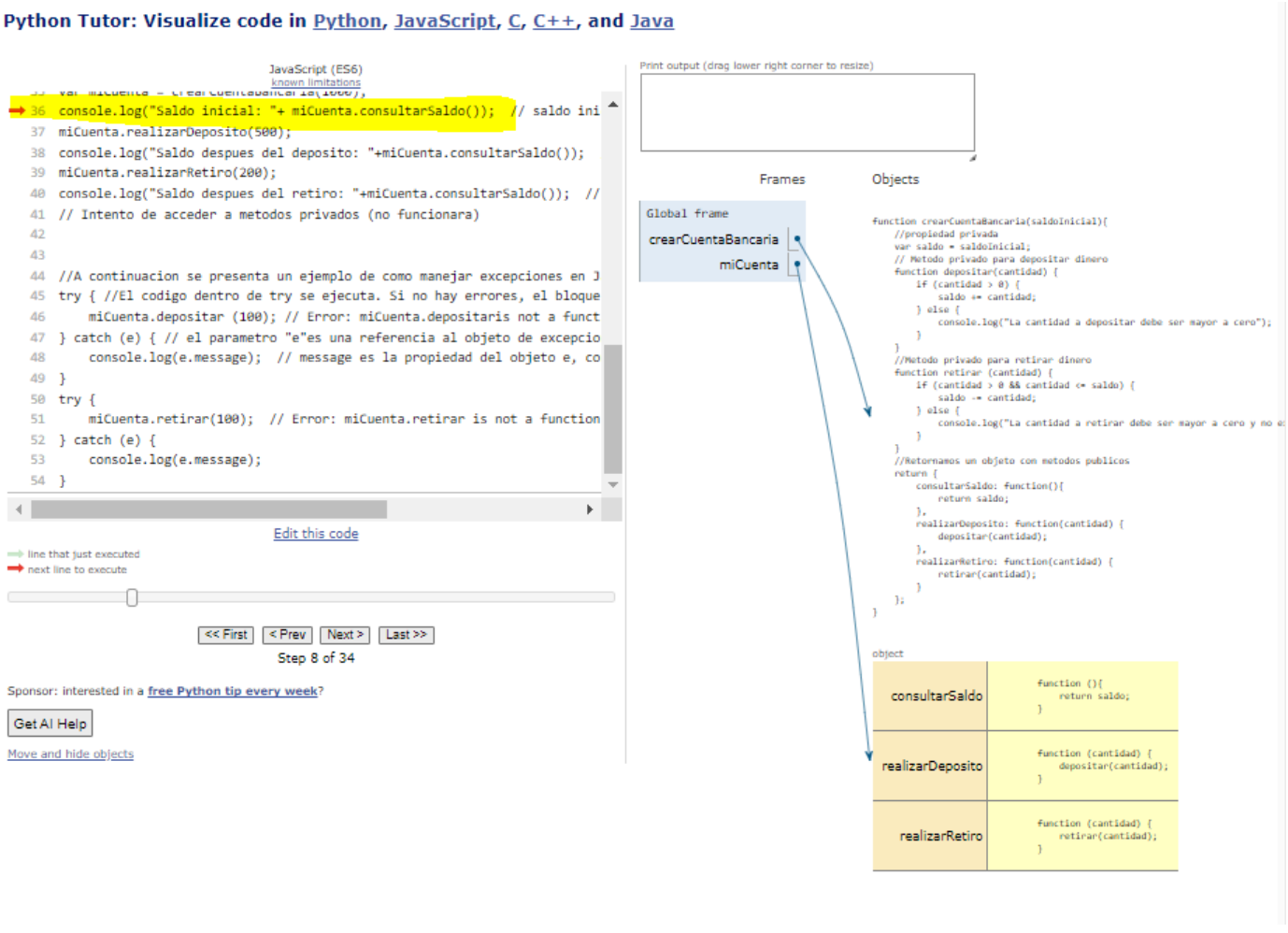
6. Iniciamos el proceso de imprimir en consola el texto “Saldo Inicial: “ + hacemos el llamado a la funcion “miCuenta.consultarSaldo()”



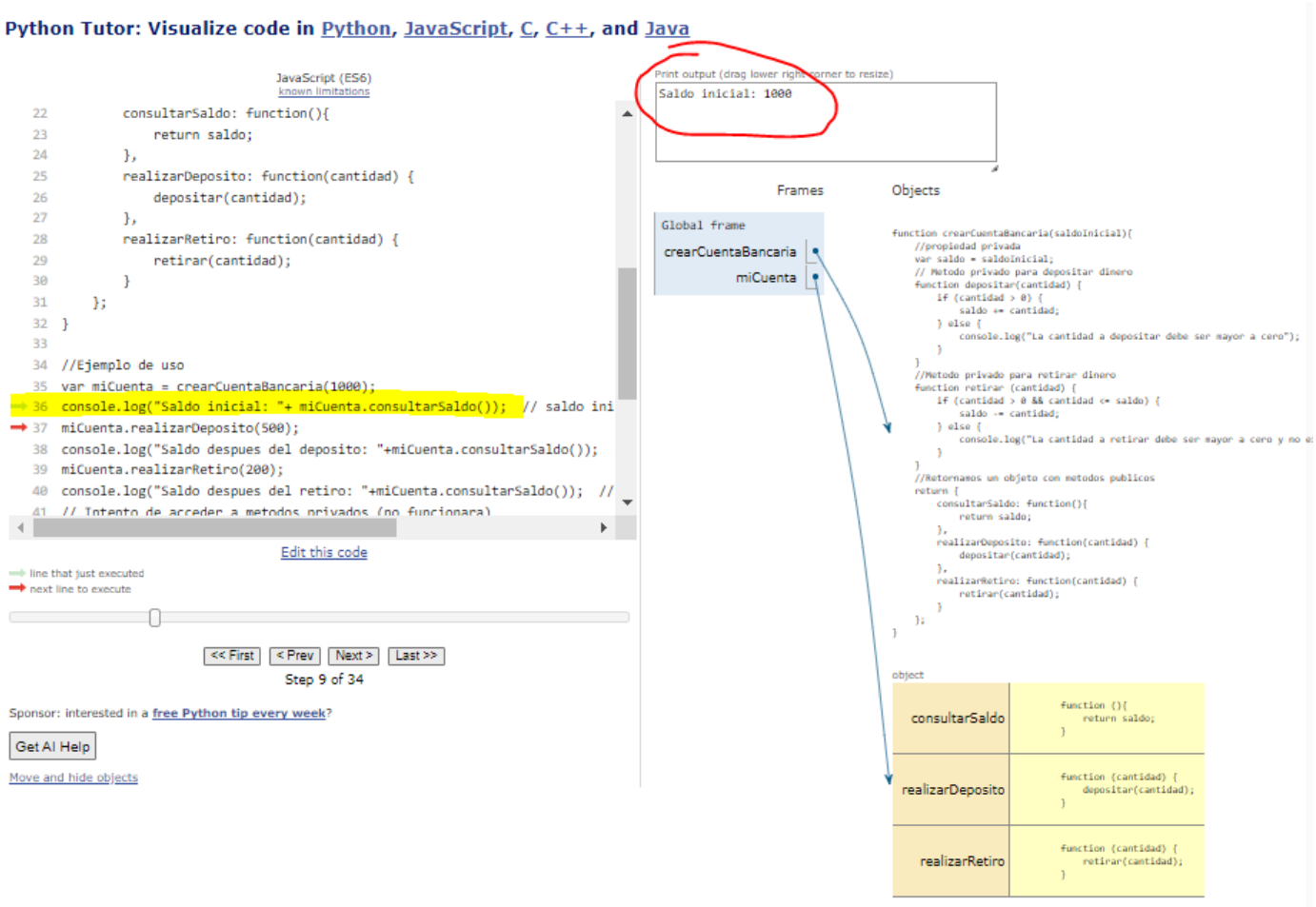
7. Al ejecutar la función "miCuenta.consultarSaldo()" va a retornar el valor "saldo" que en este momento es igual a "saldoInicial" el cual fue establecido como 1000, por lo tanto retornará el valor de 1000 como resultado de esta función.



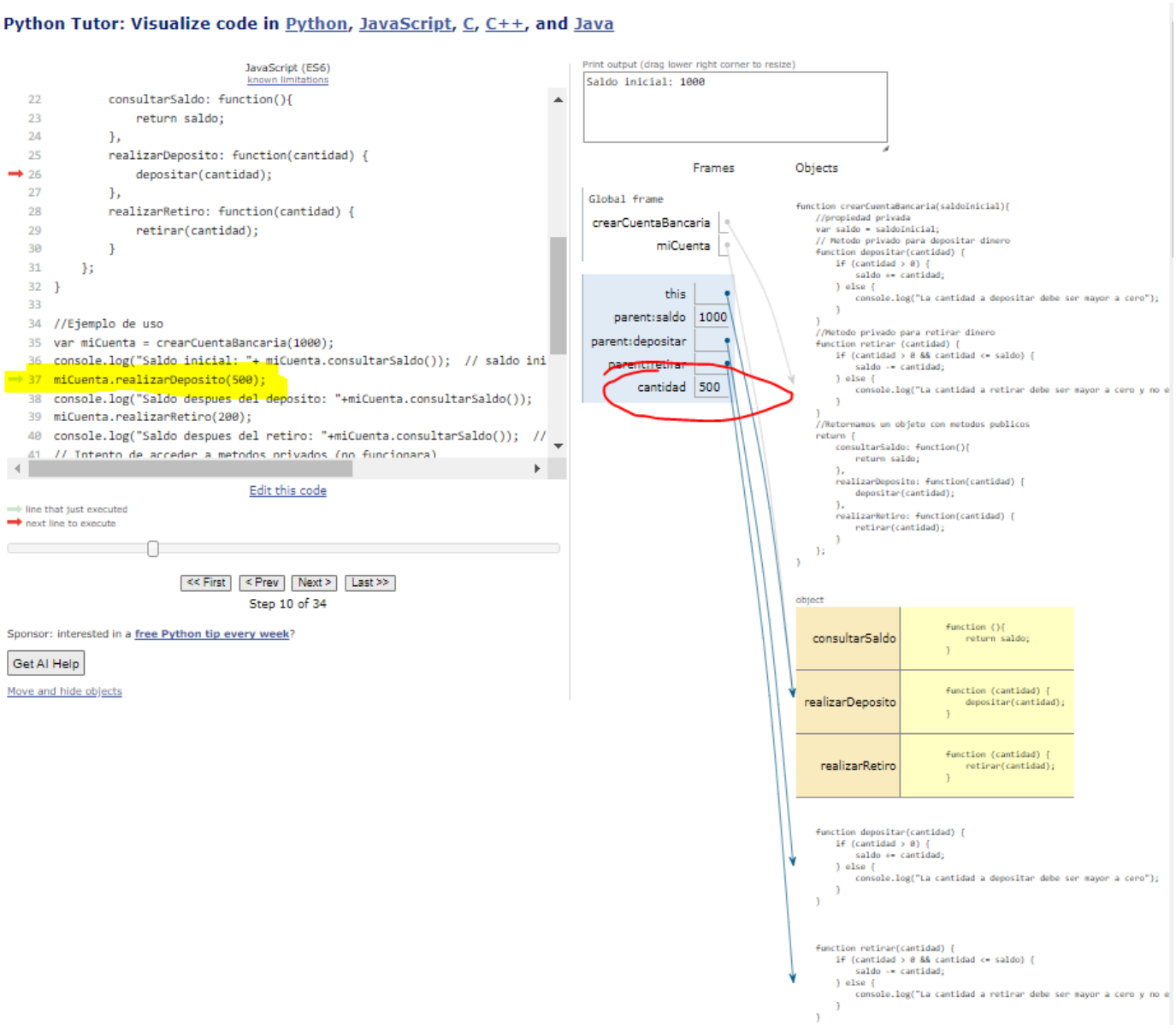
8. Se completa el recorrido del llamado de la función "miCuenta.consultarSaldo()" para poder completar el proceso de console.log.



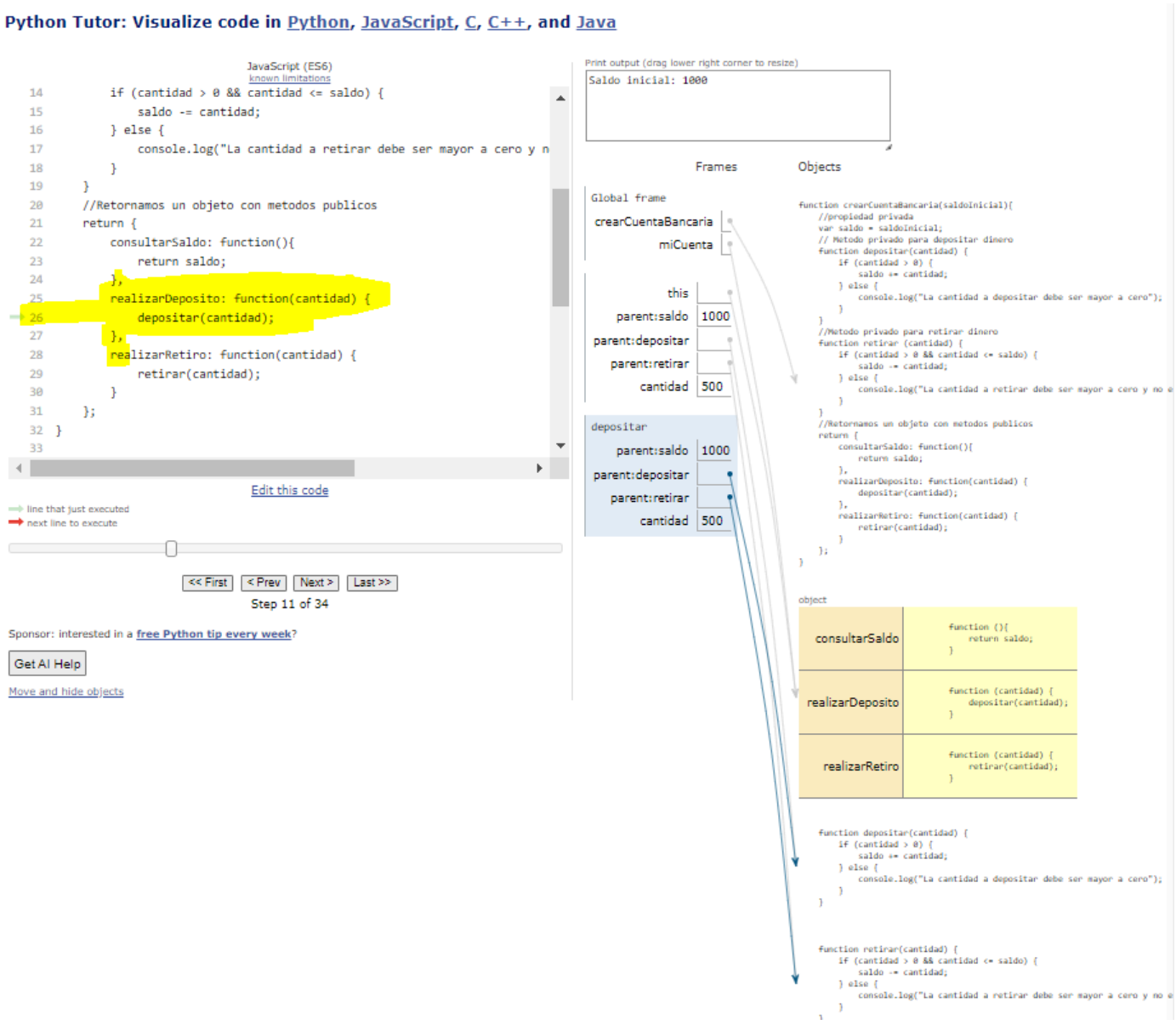
9. Imprime el resultado del código de impresión en consola imprimiendo el mensaje ya interpretado como “Saldo inicial: 1000”



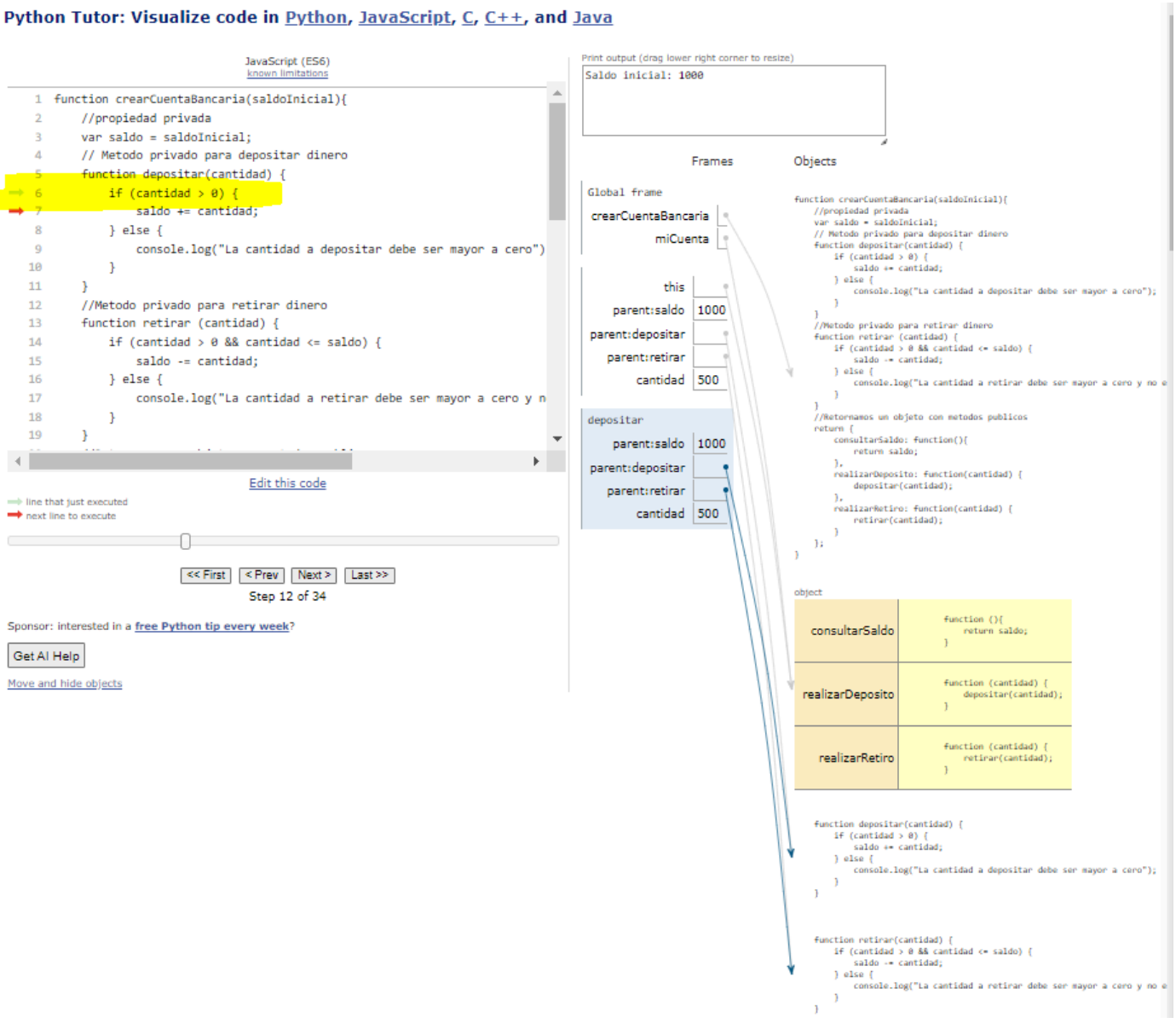
10. Ejecutamos la linea de codigo "miCuenta.realizarDeposito(500);", la cual hace el llamado a iniciar la ejecución de esa función para procesar un deposito con el valor de 500.



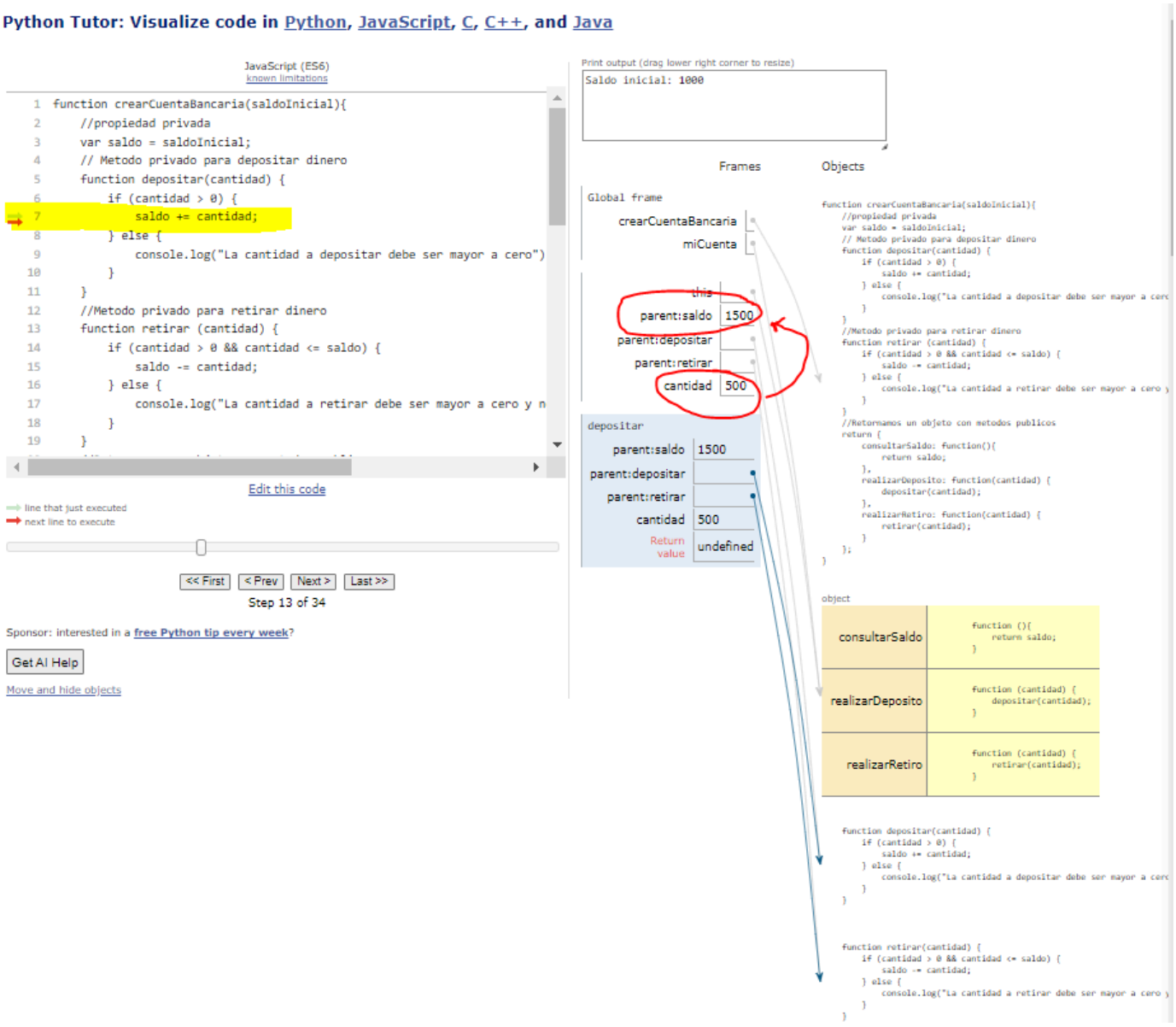
11. La función es llamada en la etapa de "return" por lo cual se empieza a ejecutar para poder regresar el valor del depósito realizado.



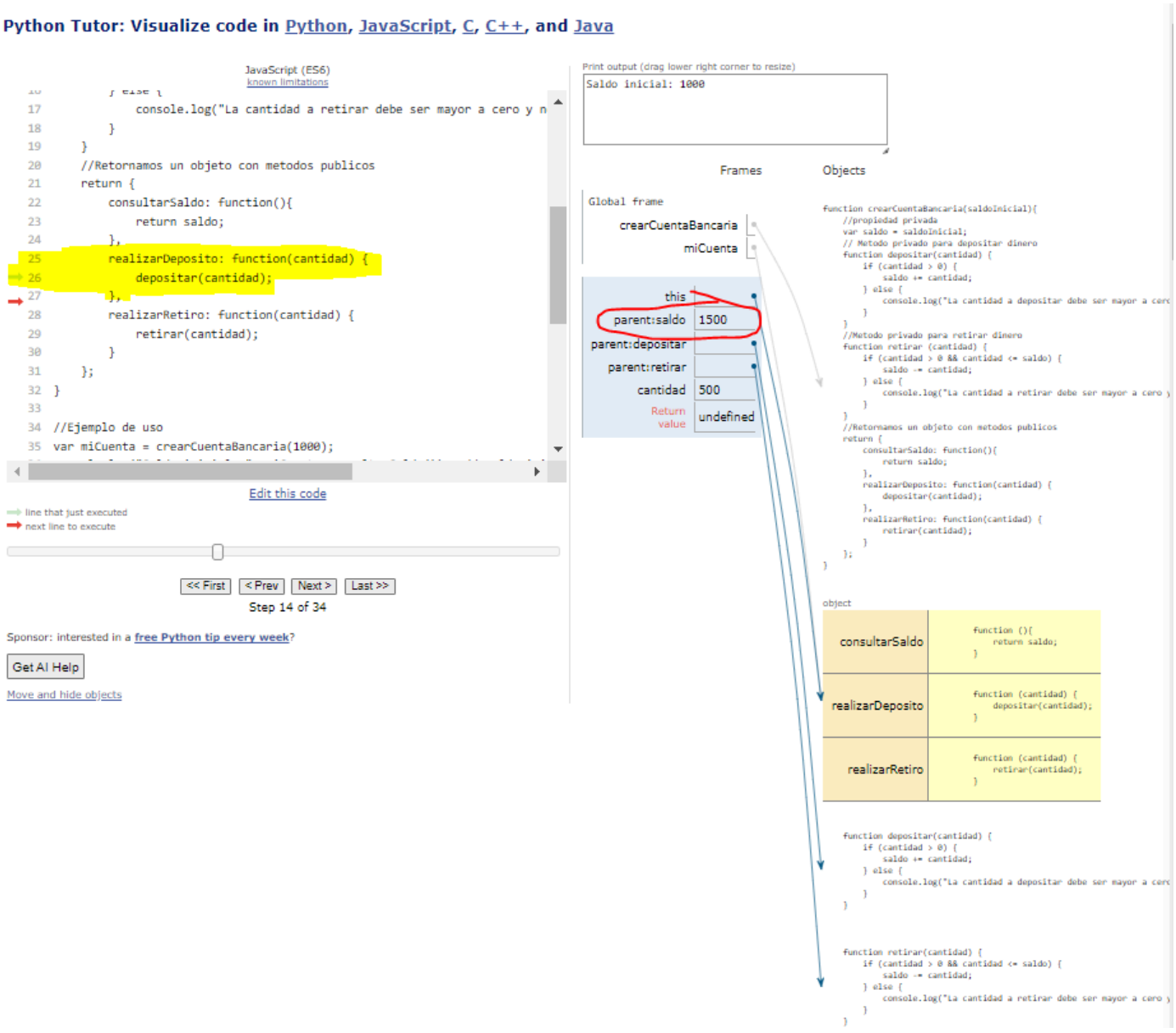
12. Al iniciar la función "depositar (cantidad)", el primer paso sería evaluar la condición que en este caso es, si el valor "cantidad" es mayor que 0, lo cual en este caso es true.



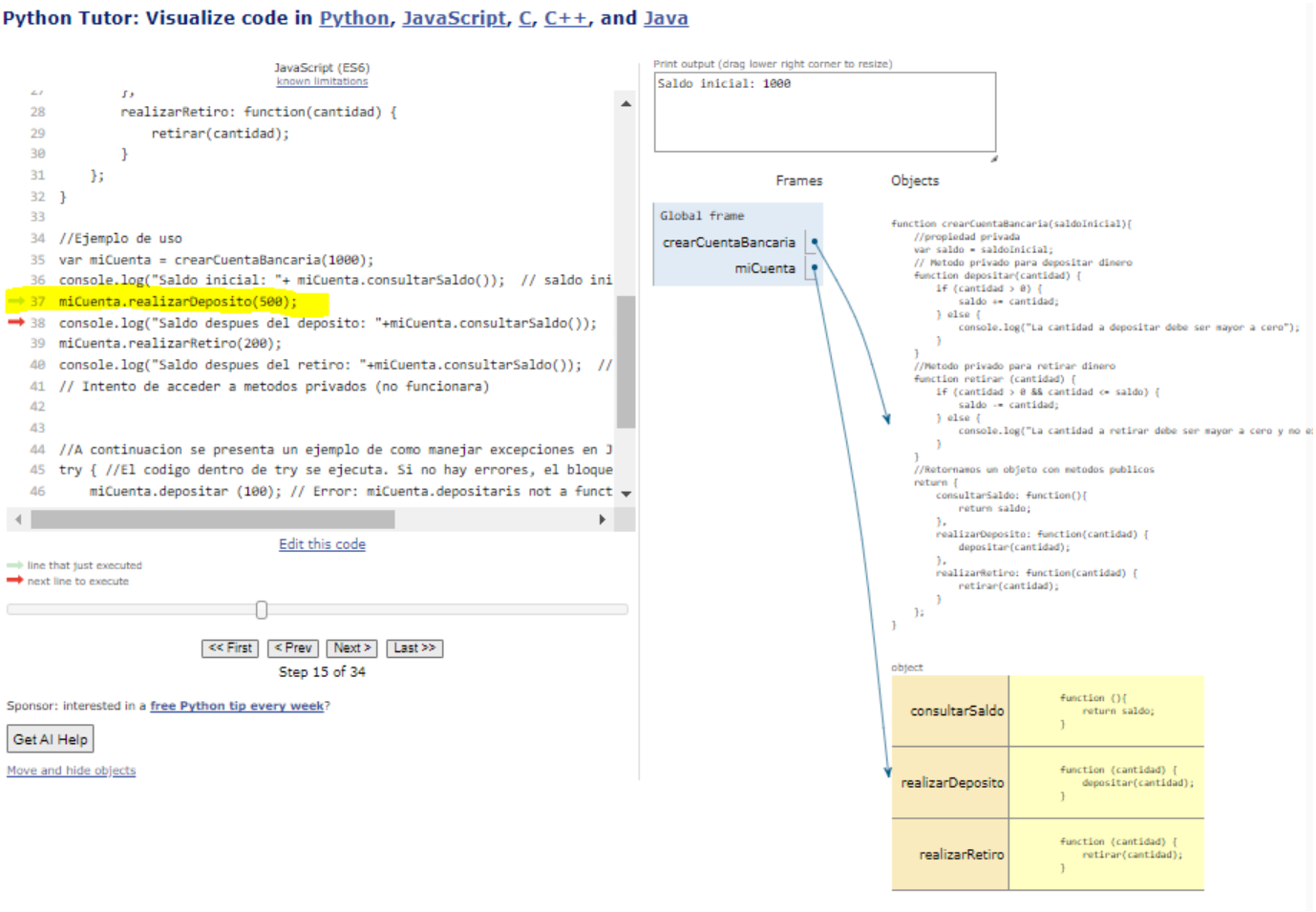
13. Debido a que la condición es true, procedemos a realizar la siguiente instrucción dentro del flujo "if" a lo cual al "saldo" actual (que es 1000) se le suma (+=) la "cantidad" ingresada, que en este caso es 500, lo cual resulta en un total de 1500.



14. Se termina de ejecutar la función por lo cual ahora se procede a registrar el valor final que será el valor regresado por la función madre.

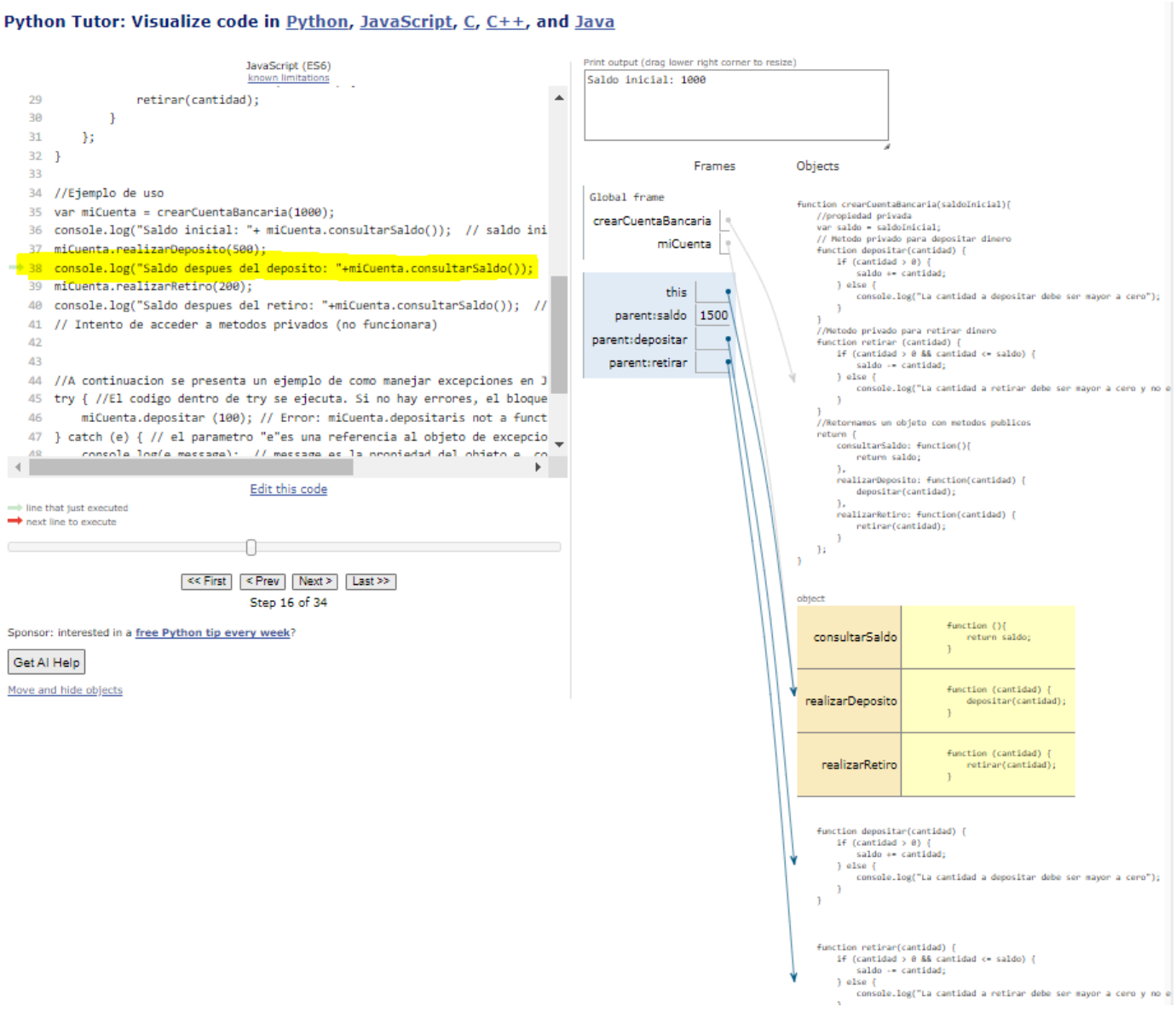


15. El llamado a la función "miCuenta.realizarDeposito(500)" ha sido completada por lo cual terminamos de recorrer esta línea de

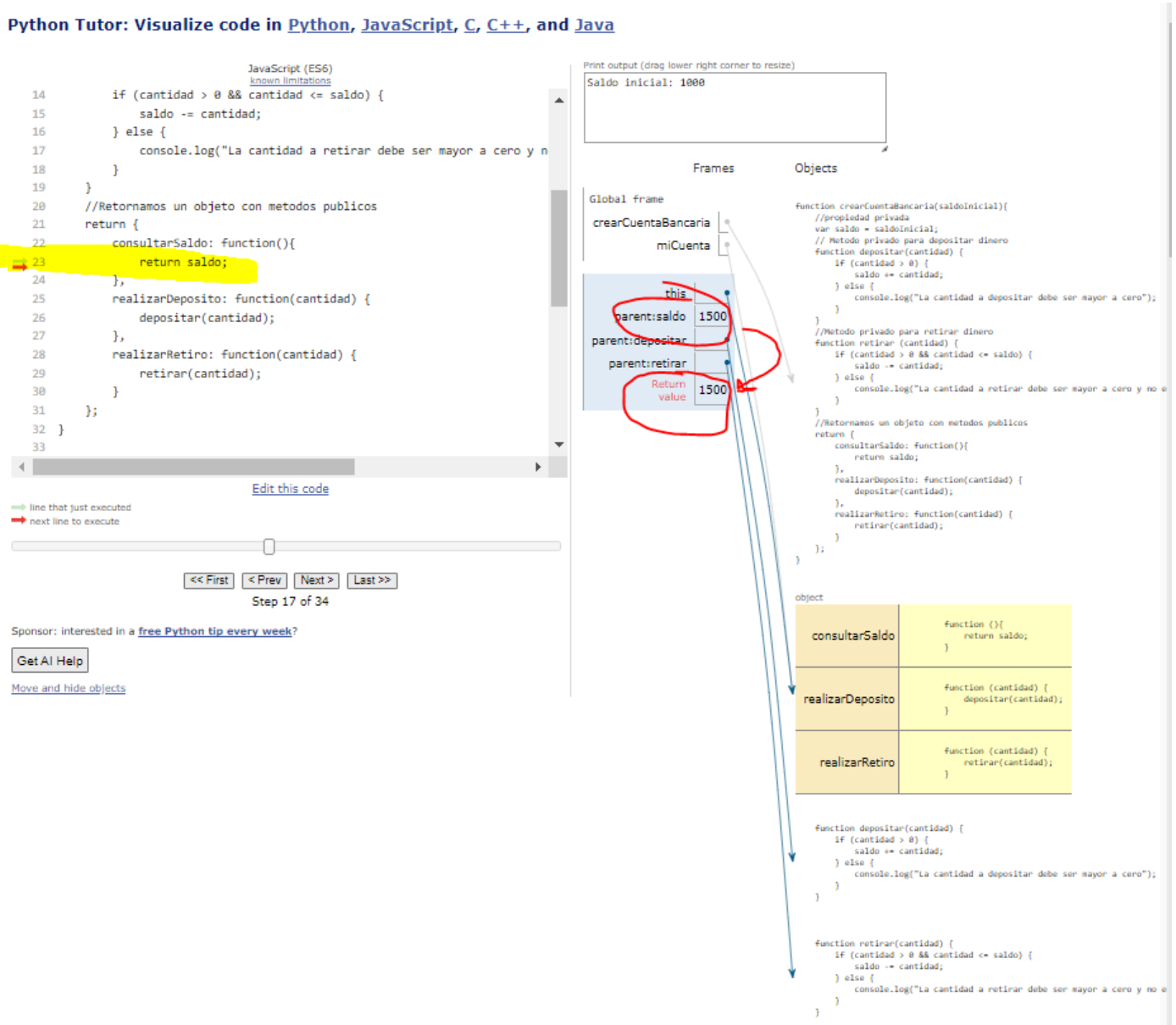


código.

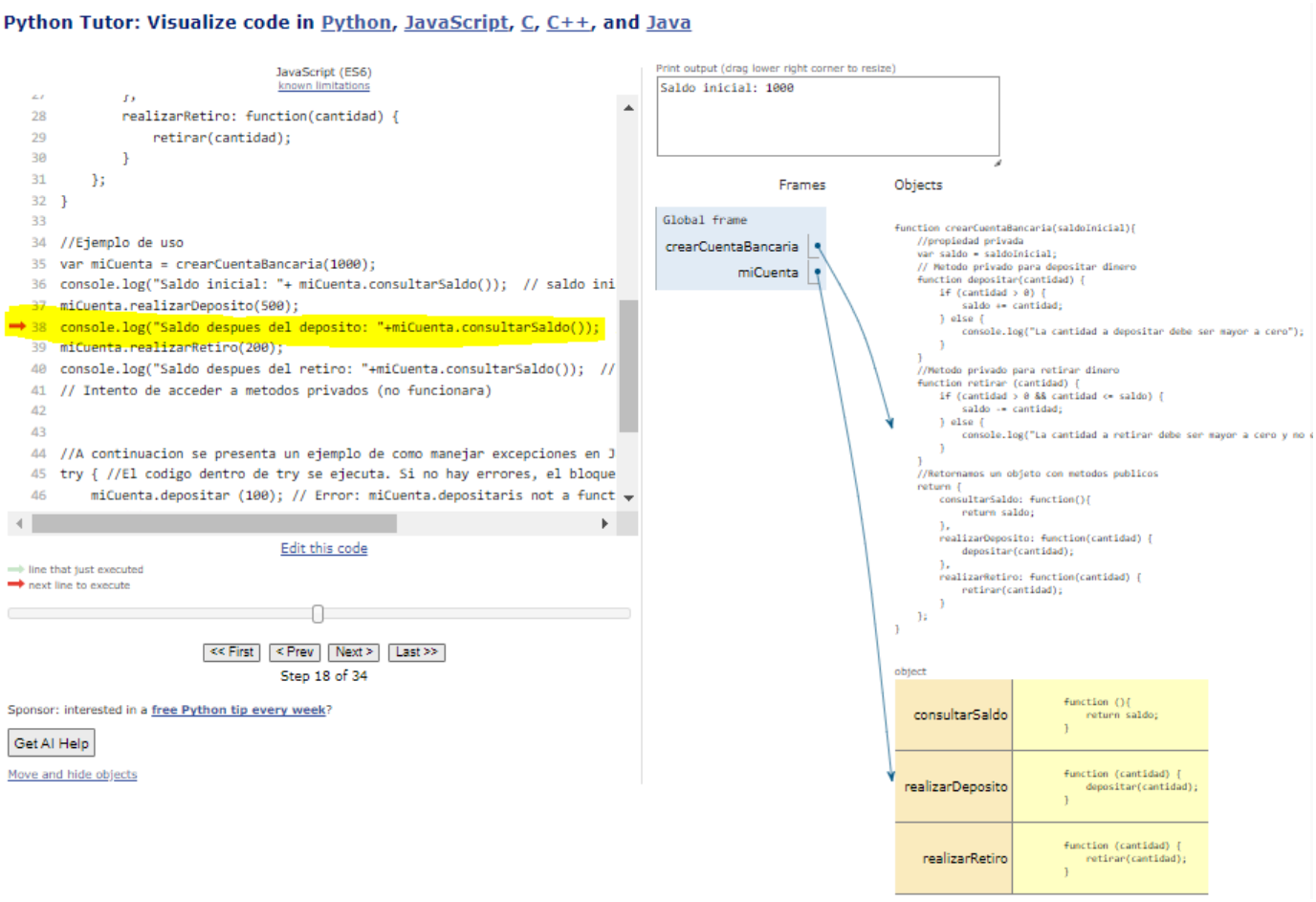
16. Procedemos a iniciar el recorrido de la instrucción **"console.log("Saldo después del depósito: "+ miCuenta.consultarSaldo());"** ante el cual se hace el llamado a la función para consultar el nuevo saldo.



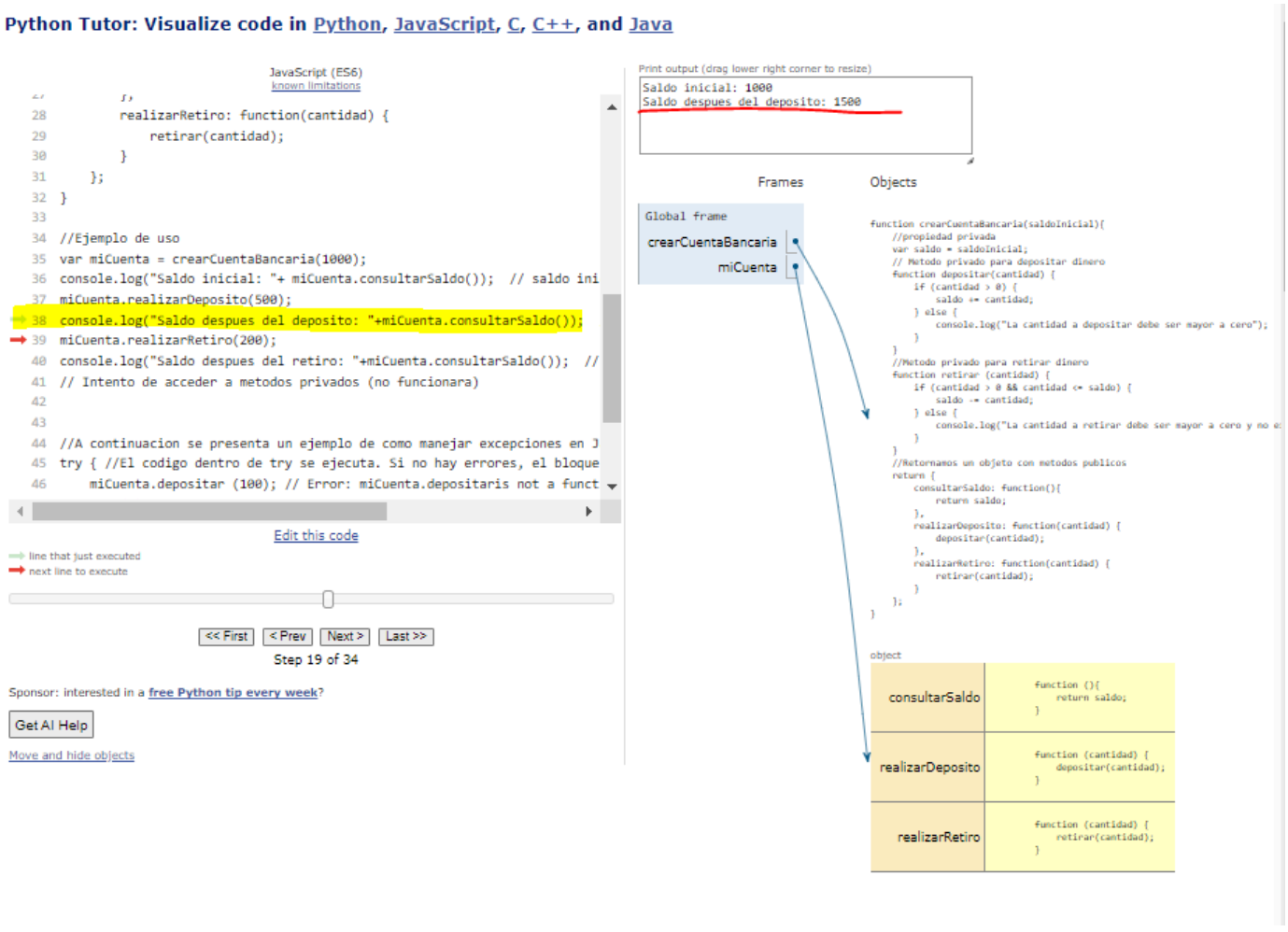
17. Ejecutamos el llamado a la función la cual retorna el valor de "saldo" que en este momento es el valor de 1500.



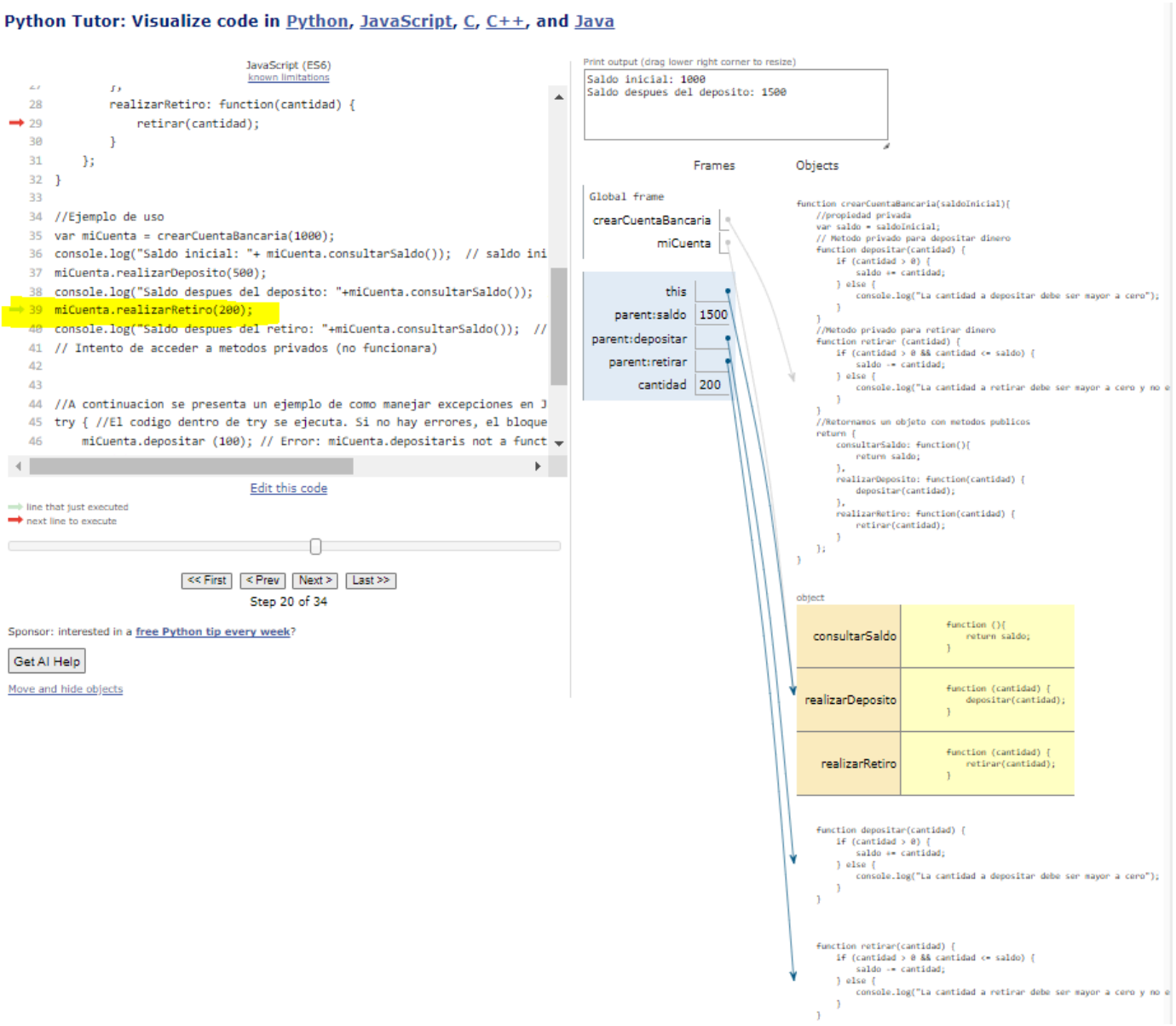
18. Se termina de recorrer la solicitud de impresión en consola y termina de recorrer la funcion **miCuenta.consultarSaldo()**



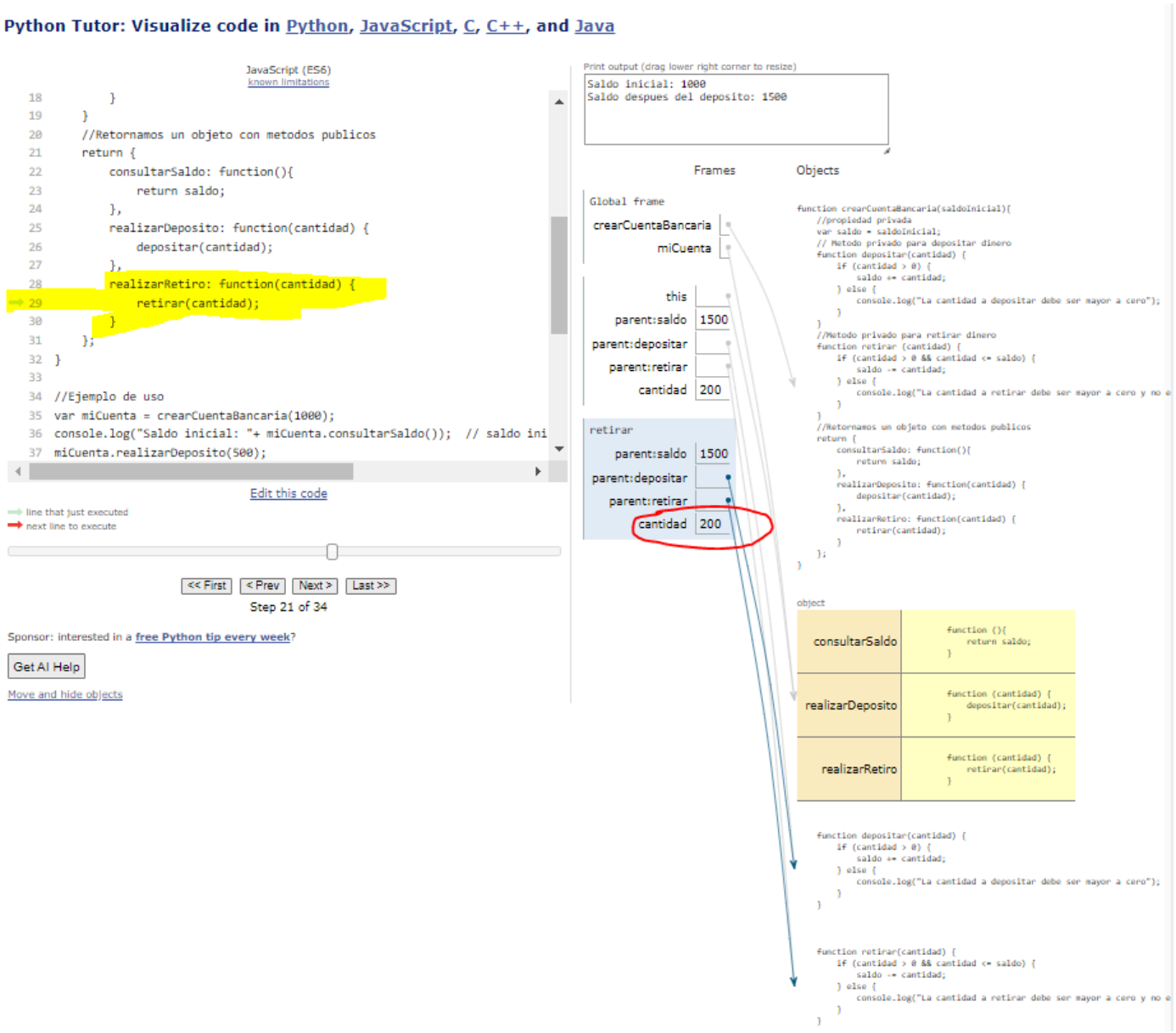
19. Imprime el resultado de la solicitud de imprimir en consola.



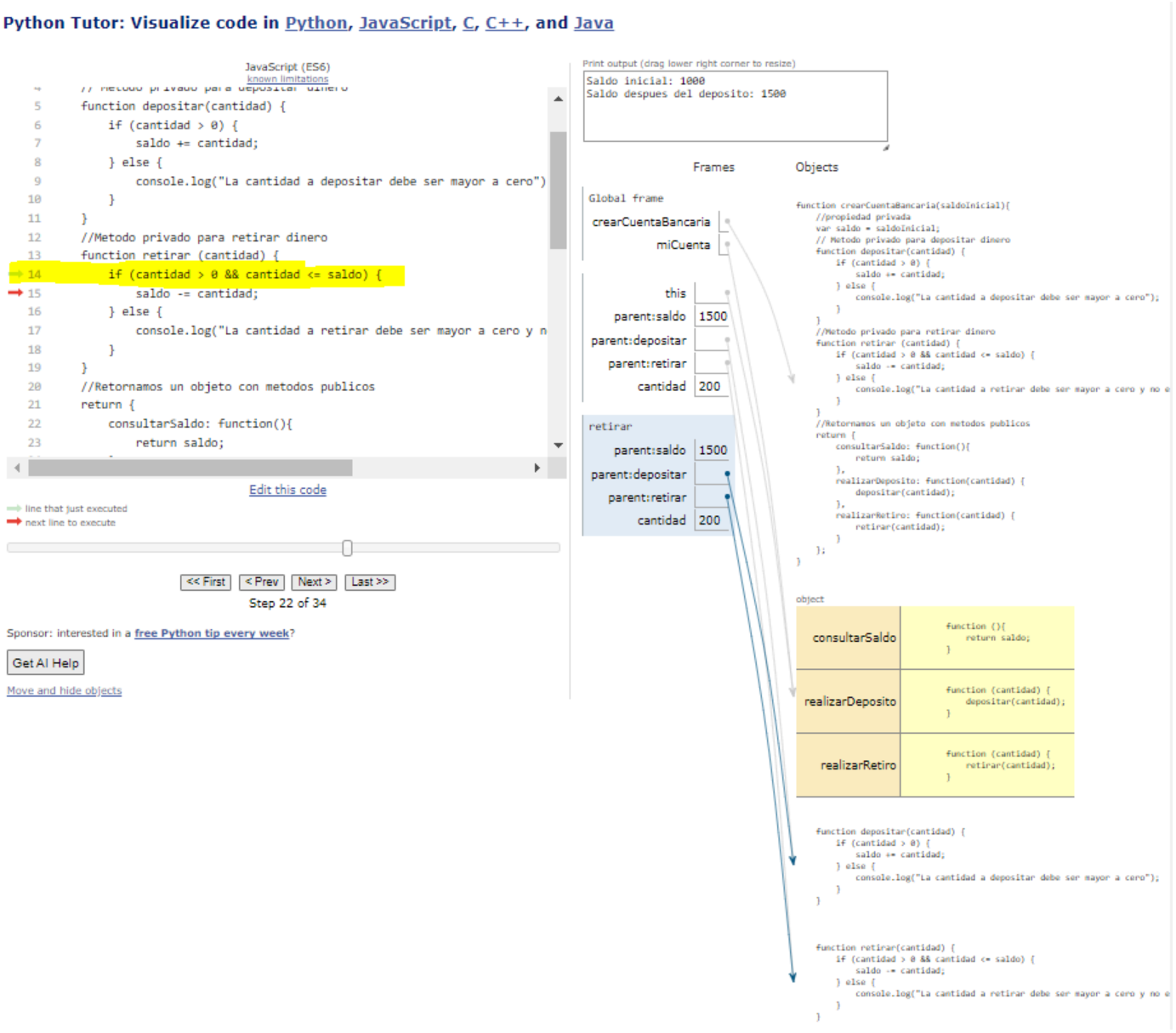
20. Se hace la petición de hacer un retiro de 200, haciendo el llamado de la función "miCuenta.realizarRetiro(200);"



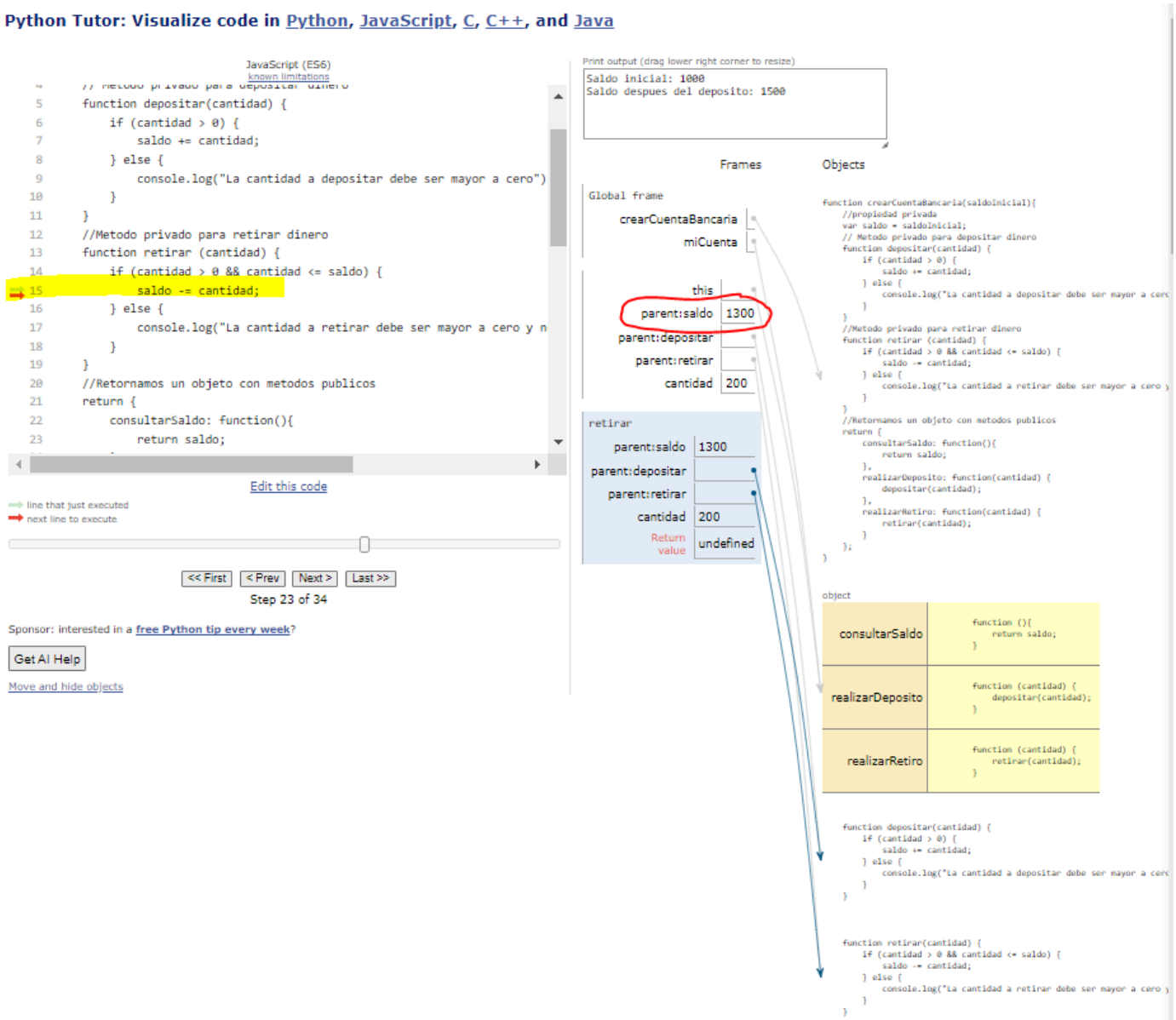
21. Se inicia la ejecución de la función de retiro por la cantidad establecida de 200



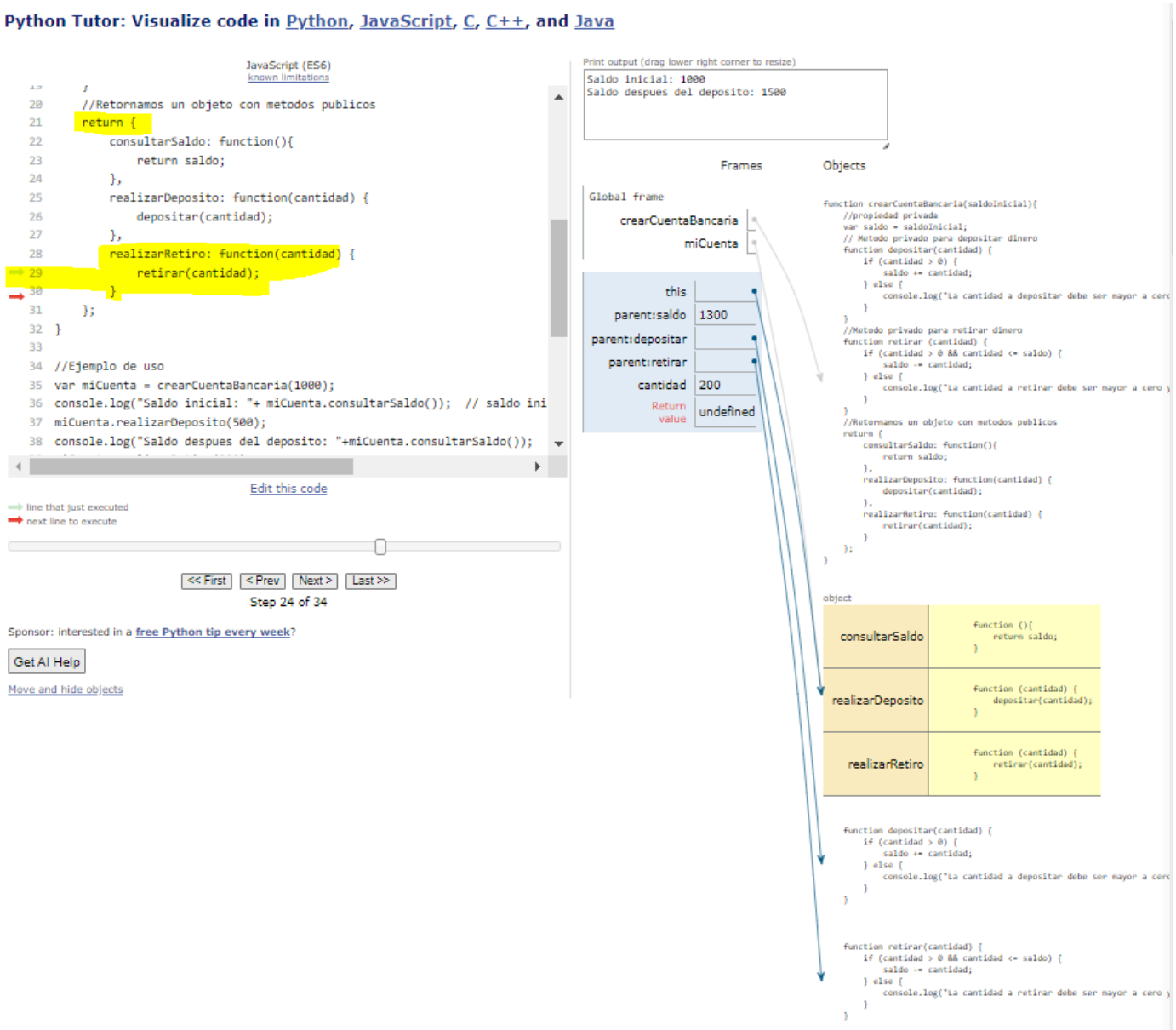
22. Comenzamos evaluando la condición **"if (cantidad > 0 && cantidad <= saldo)"** ante lo cual es true en ambas condiciones.



23. Ya que la condición "if" es true, procedemos a ejecutar la instrucción de que "saldo" (actualmente 1500) se le resta (-=) la "cantidad" ingresada que es 200. Por lo cual "saldo" tiene un nuevo valor de 1300.



24. Se termina de ejecutar la función de retiro ante lo cual se vuelve un valor de “return” para proceder con el recorrido del código.



25. Se termina de ejecutar el recorrido del llamado a la función de realizarRetiro por un valor de 200.

Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

JavaScript (ES6)
known limitations

```
30          }
31      };
32  }
33
34  //Ejemplo de uso
35  var miCuenta = crearCuentaBancaria(1000);
36  console.log("Saldo inicial: "+ miCuenta.consultarSaldo());  // saldo ini
37  miCuenta.realizarDeposito(500);
38  console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39  miCuenta.realizarRetiro(200);
40  console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo());  //
41  // Intento de acceder a metodos privados (no funcionara)
42
43
44  //A continuacion se presenta un ejemplo de como manejar excepciones en J
45  try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46      miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
47  } catch (e) { // el parametro "e"es una referencia al objeto de excepcio
48      console.log(e.message);  // message es la propiedad del objeto e, co
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 25 of 34

Sponsor: interested in a free Python tip every week?

Get AI Help

Move and hide objects

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo despues del deposito: 1500
```

Frames | Objects

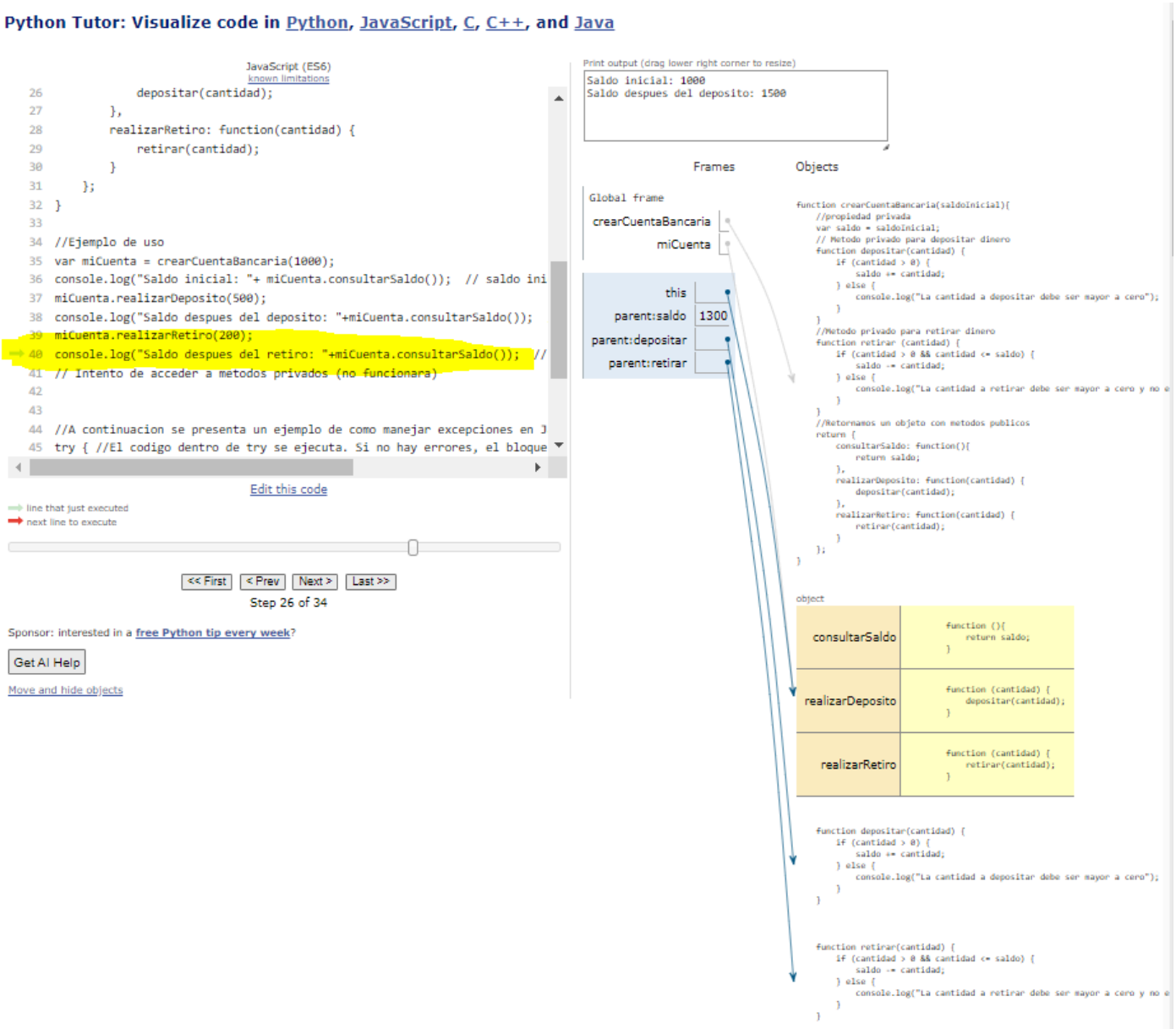
Global frame
crearCuentaBancaria
miCuenta

```
function crearCuentaBancaria(saldoInicial){
    //propiedad privada
    var saldo = saldoInicial;
    // Metodo privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero");
        }
    }
    //Metodo privado para retirar dinero
    function retirar (cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no e
        }
    }
    //Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function(){
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

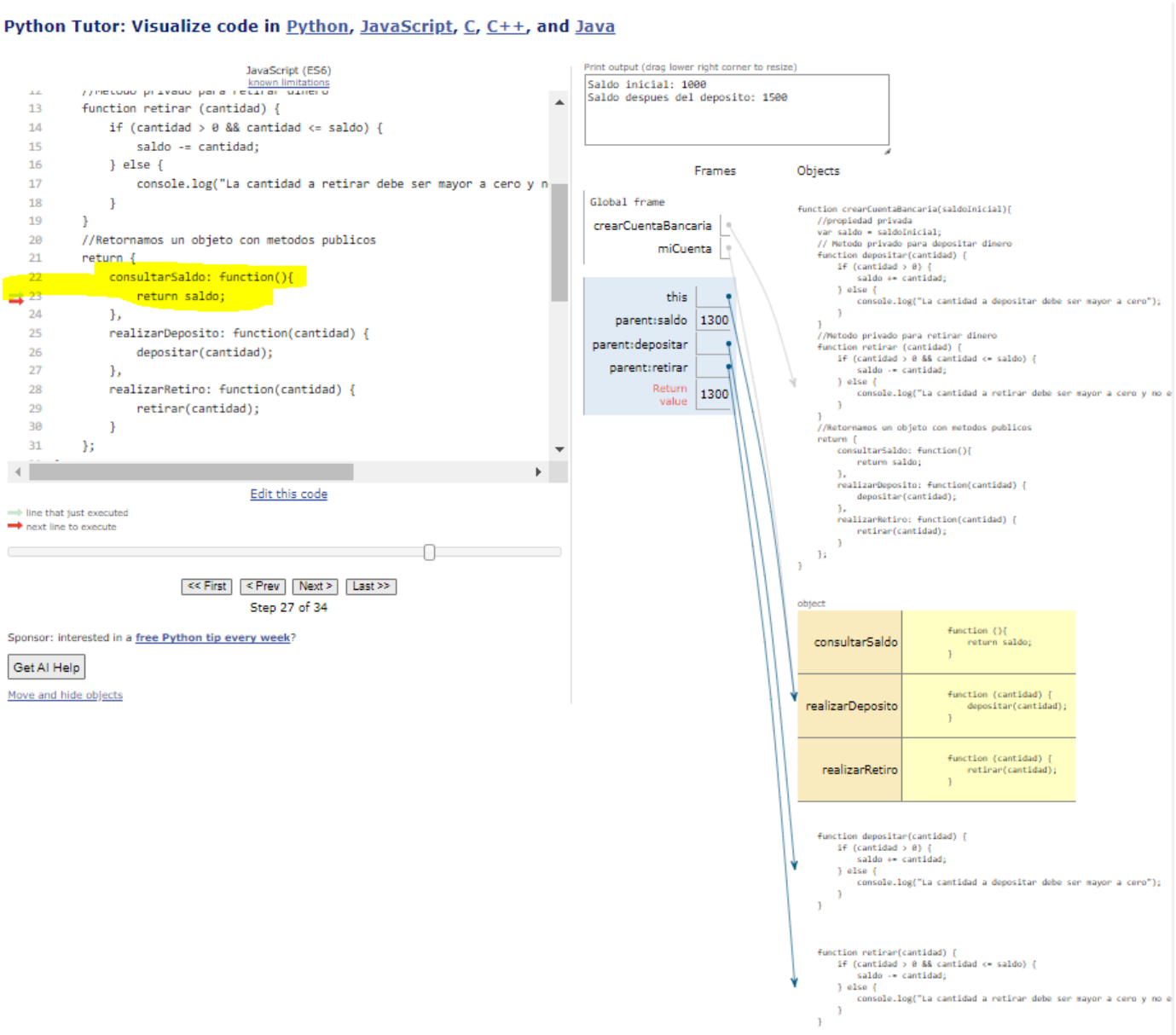
object

consultarSaldo	function (){ return saldo; }
realizarDeposito	function (cantidad) { depositar(cantidad); }
realizarRetiro	function (cantidad) { retirar(cantidad); }

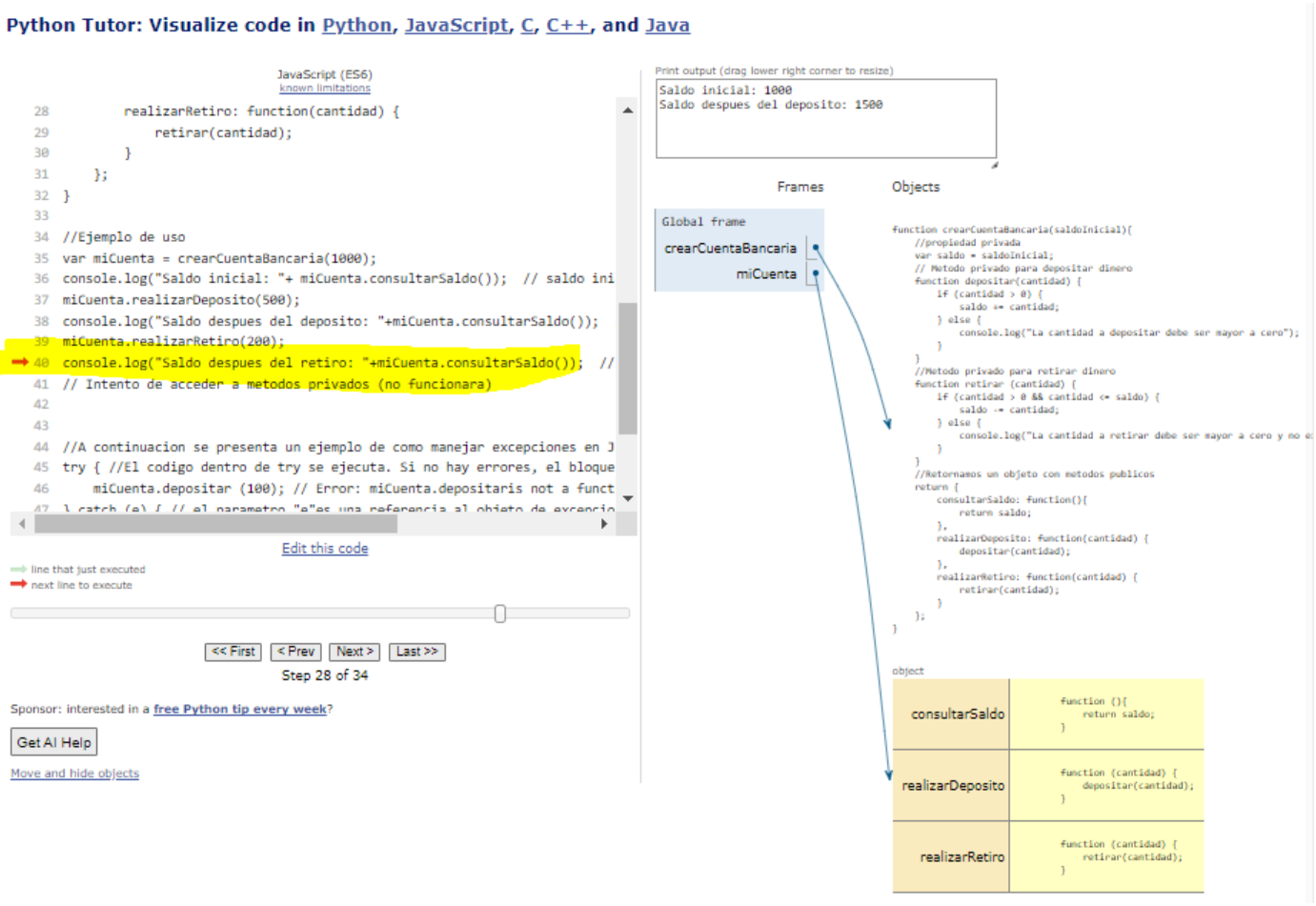
26. Procedemos a iniciar la petición de imprimir en consola el mensaje de saldo después del retiro, haciendo el llamado a la funcion **"consultarSaldo"**



27. Se inicia el recorrido de la funcion **"consultarSaldo"** ante el cual esperamos retornar el nuevo "saldo"



28. Ejecutamos la funcion “consultarSaldo” la cual indica que retornaremos el valor de “saldo” que actualmente es 1300.



29. Ya que completamos todas las peticiones y recorridos para obtener el valor que queremos imprimir, procedemos a realizar la impresión en consola.

Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

JavaScript (ES6)
known limitations

```
28          realizarRetiro: function(cantidad) {
29              retirar(cantidad);
30          }
31      };
32  }
33
34  //Ejemplo de uso
35  var miCuenta = crearCuentaBancaria(1000);
36  console.log("Saldo inicial: "+ miCuenta.consultarSaldo());  // saldo ini
37  miCuenta.realizarDeposito(500);
38  console.log("Saldo despues del deposito: "+miCuenta.consultarSaldo());
39  miCuenta.realizarRetiro(200);
40  console.log("Saldo despues del retiro: "+miCuenta.consultarSaldo());  //
41  // Intento de acceder a metodos privados (no funcionara)
42
43
44  //A continuacion se presenta un ejemplo de como manejar excepciones en J
45  try { //El codigo dentro de try se ejecuta. Si no hay errores, el bloque
46      miCuenta.depositar (100); // Error: miCuenta.depositaris not a funct
47  } catch (e) { // el parametro "e"es una referencia al objeto de excepcio
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 29 of 34

Sponsor: interested in a free Python tip every week?

Get AI Help

Move and hide objects

Print output (drag lower right corner to resize)

```
Saldo inicial: 1000
Saldo despues del deposito: 1500
Saldo despues del retiro: 1300
```

Frames

Global frame
- crearCuentaBancaria
- miCuenta

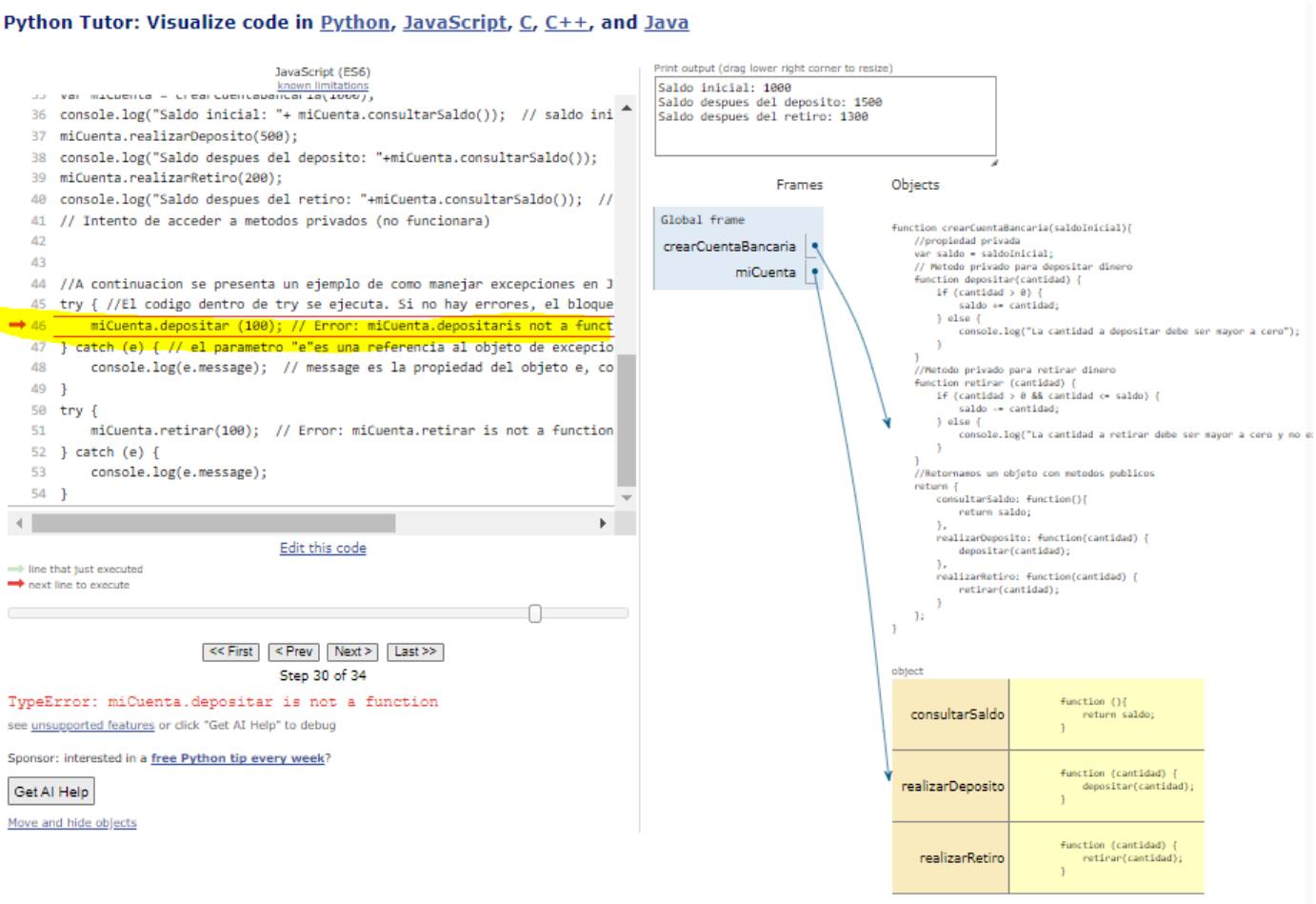
Objects

```
function crearCuentaBancaria(saldoInicial){
    //propiedad privada
    var saldo = saldoInicial;
    // Metodo privado para depositar dinero
    function depositar(cantidad) {
        if (cantidad > 0) {
            saldo += cantidad;
        } else {
            console.log("La cantidad a depositar debe ser mayor a cero");
        }
    }
    //Metodo privado para retirar dinero
    function retirar (cantidad) {
        if (cantidad > 0 && cantidad <= saldo) {
            saldo -= cantidad;
        } else {
            console.log("La cantidad a retirar debe ser mayor a cero y no e
        }
    }
    //Retornamos un objeto con metodos publicos
    return {
        consultarSaldo: function(){
            return saldo;
        },
        realizarDeposito: function(cantidad) {
            depositar(cantidad);
        },
        realizarRetiro: function(cantidad) {
            retirar(cantidad);
        }
    };
}
```

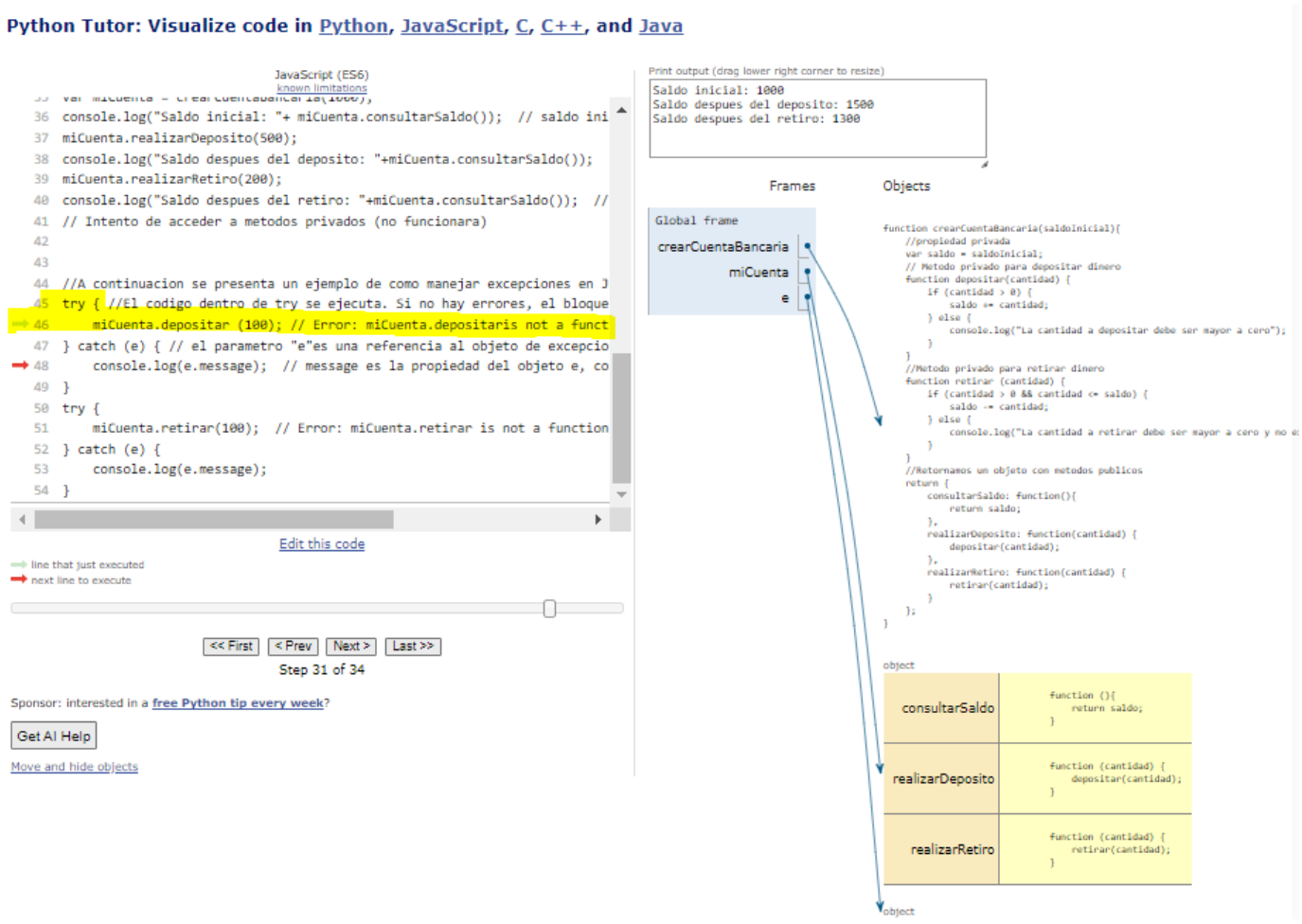
object

consultarSaldo	function (){ return saldo; }
realizarDeposito	function (cantidad) { depositar(cantidad); }
realizarRetiro	function (cantidad) { retirar(cantidad); }

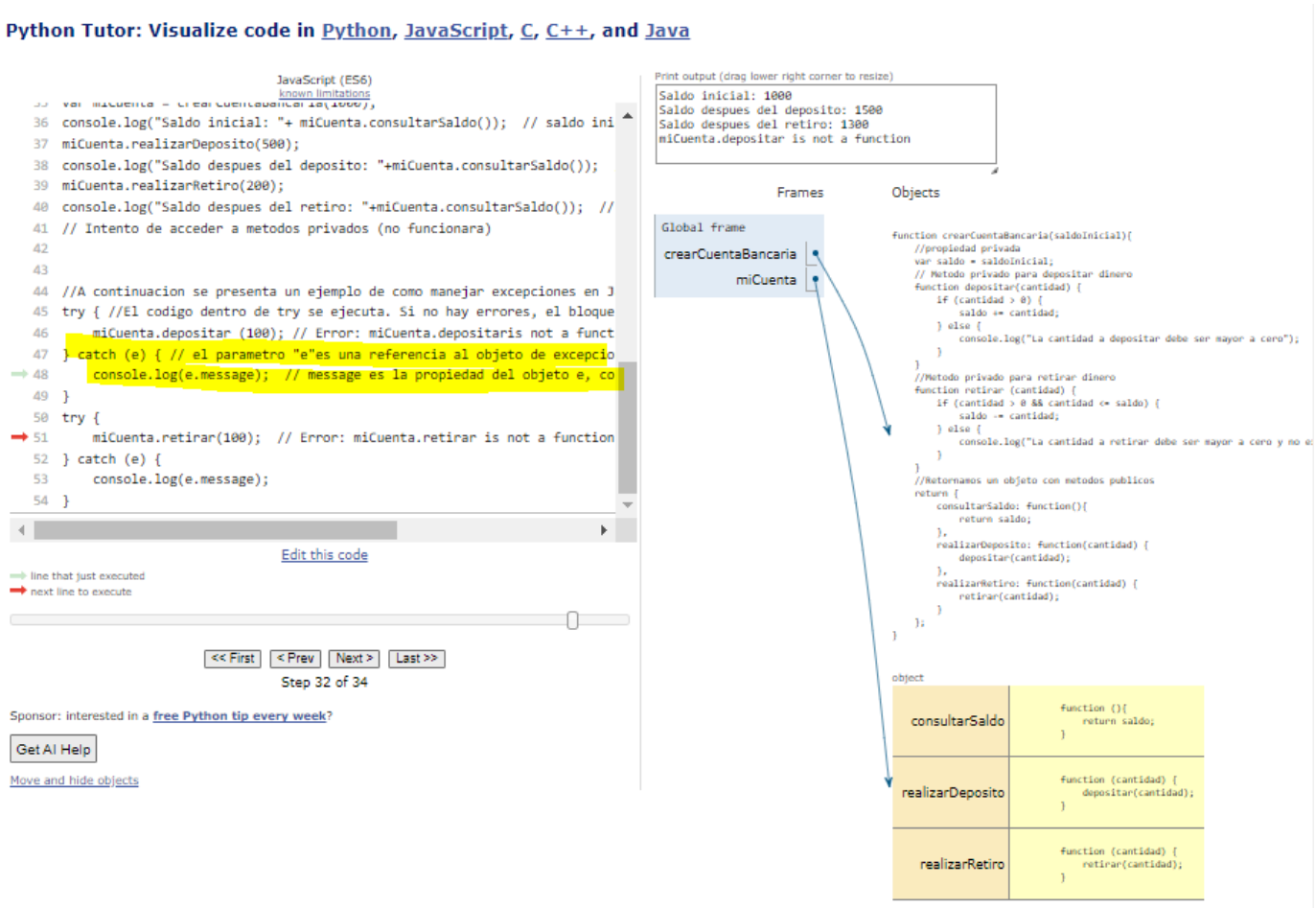
30. **miCuenta.depositar (100)** no es una función por lo que la consola nos muestra el error. Esto lo intentamos dentro de un "try" para poder ejemplificar cómo poder manejar este tipo de situaciones.



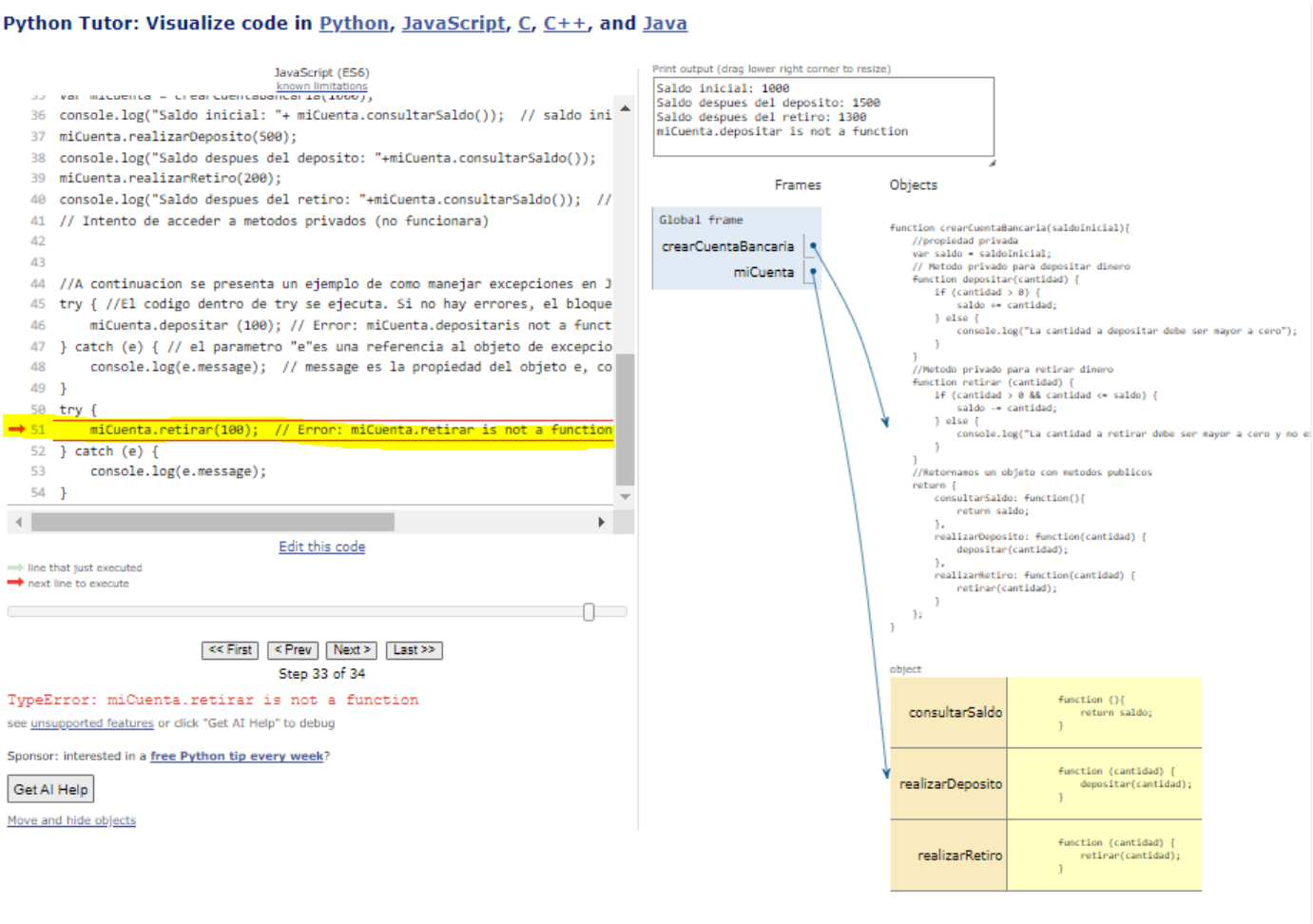
31. Ya que el "try" nos generó un error, procederemos con el siguiente paso.



32. Procedemos a ejecutar el "catch", por lo cual intentamos mostrar al usuario qué tipo de error estamos enfrentando



33. “Catch” nos solicita registrar el tipo de error que estamos enfrentando, el cual en este caso es el siguiente:



34. Procedemos a completar la impresión en consola del mensaje solicitado del error que hemos enfrentado, y con esto finalizamos nuestra aplicación.