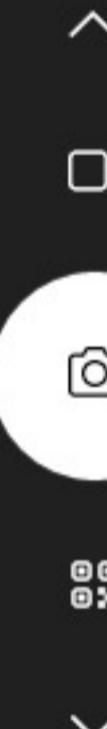


ATRIBUTOS CONSTRUCTOR Y METODOS.py

```
1 class Celular:
2     def __init__(self, marca, modelo, camara):
3         self.marca = marca
4         self.modelo = modelo
5         self.camara = camara
6
7     def llamar(self):
8         print(f'estas llamando desde tu {self.marca} {self.modelo}')
9
10    def cortar(self):
11        print(f'cortaste la llamada desde tu {self.marca} {self.modelo}')
12
13 celular1 = Celular("Iphone ","Iphone 15", "48MP")
14 celular2 = Celular("Samsung ","S23", "54MP")
15
16 celular1.llamar()
17 celular1.cortar()
18
19 celular2.llamar()
20 celular2.cortar()
21 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/ATRIBUTOS CONSTRUCTOR Y METODOS.py"
estas llamando desde tu Iphone Iphone 15
cortaste la llamada desde tu Iphone Iphone 15
estas llamando desde tu Samsung S23
cortaste la llamada desde tu Samsung S23
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```



The image shows a split-screen interface of Visual Studio Code. On the left side, the code editor displays a Python script named `EJERCICIO1.py`. The code defines a class `Estudiante` with an `__init__` method and creates an instance `Fabricio` with name "Fabricio", age 30, and semester 2. It then prints the name. The terminal below shows the script running successfully, outputting "Fabricio". On the right side, there is a camera feed window titled "Cámaras" showing a man with glasses. A blue circle highlights his face. The camera interface includes controls for zoom, brightness, and a camera button.

```
class Estudiante:
    def __init__(self,nombre,edad,semestre):
        self.nombre = nombre
        self.edad = edad
        self.semestre = semestre

Fabricio = Estudiante("Fabricio",30,2)

print(Fabricio.nombre)
```

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Ejercicio 1.py"
Fabricio
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```

Visual Studio Code

EJERCICIO 1.1.py

```
1 class Estudiante:
2     def __init__(self,nombre,edad,semestre):
3         self.nombre = nombre
4         self.edad = edad
5         self.semestre = semestre
6
7     nombre = input("Escriba su nombre:")
8     edad = input("Su edad:")
9     semestre = input("Ahora, su semestre:")
10
11 estudiante = Estudiante(nombre,edad, semestre)
12
13 print(f"""
14     DATOS DEL ESTUDIANTE: \n\n
15     Nombre: {estudiante.nombre} \n
16     Edad: {estudiante.edad} \n
17     Semestre: {estudiante.semestre} \n
18 """)
19
20
21 while True:
22     estudiar = input()
23     if (estudiar.lower == "estudiar"):
24         estudiante.estudiar
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Cámara

DATOS DEL ESTUDIANTE:

Nombre: Fabricio

Edad: 30

Semestre: Semestre



Visual Studio Code

HERENCIA.py

```
1 class Persona:
2     def __init__(self, nombre, edad, nacionalidad):
3         self.nombre = nombre
4         self.edad = edad
5         self.nacionalidad = nacionalidad
6
7     def hablar(self):
8         print(f"Hello, my name is {self.nombre} and I am {self.edad} years old, I am from {self.nacionalidad}.")
9
10    class Empleado(Persona):
11        def __init__(self, nombre, edad, nacionalidad, trabajo, salario):
12            super().__init__(nombre, edad, nacionalidad)
13            self.trabajo = trabajo
14            self.salario = salario
15
16
17    fabricio = Empleado("Fabricio", "30", "Ecuatoriano", "Ingeniero Electrico", 1200000)
18
19    fabricio.hablar()
20
21
22
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

Cámaras

Cámara



PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Herencia.py"
Hello, my name is Fabricio and I am 30 years old, I am from Ecuatoriano.
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Visual Studio Code
- File Explorer:** Shows a file named "HERENCIA SIMPLE.py".
- Search Bar:** A magnifying glass icon.
- Code Editor:** Displays the following Python code:

```
4     self.edad = edad
5     self.nacionalidad = nacionalidad
6
7     def hablar(self):
8         print(f"Buenas noches {self.nombre} y estoy conversando mucho")
9
10    class Empleado(Persona):
11        def __init__(self, nombre, edad, nacionalidad, trabajo, salario):
12            super().__init__(nombre, edad, nacionalidad)
13            self.trabajo = trabajo
14            self.salario = salario
15
16    Fabricio = Empleado("Fabricio", "30", "Ecuatoriano", "Ingeniero", "2300 do
17    Fabricio.hablar()
18
19    class Estudiante(Persona):
20        def __init__(self, nombre, edad, nacionalidad, universidad, carrera):
21            super().__init__(nombre, edad, nacionalidad)
22            self.universidad = universidad
23            self.carrera = carrera
24
25    Fernando = Estudiante("Fernando", "30", "Ecuatoriano", "ESPE", "TICS")
26    print("El estudiante", Fernando.nombre, "tiene", Fernando.edad, "años, est
27    Fernando.hablar()
```

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Herencia Simple.py"
Buenas noches Fabricio y estoy conversando mucho
El estudiante Fernando tiene 30 años, estudia en la ESPE en la carrera TICS
Buenas noches Fernando y estoy conversando mucho
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```



Visual Studio Code

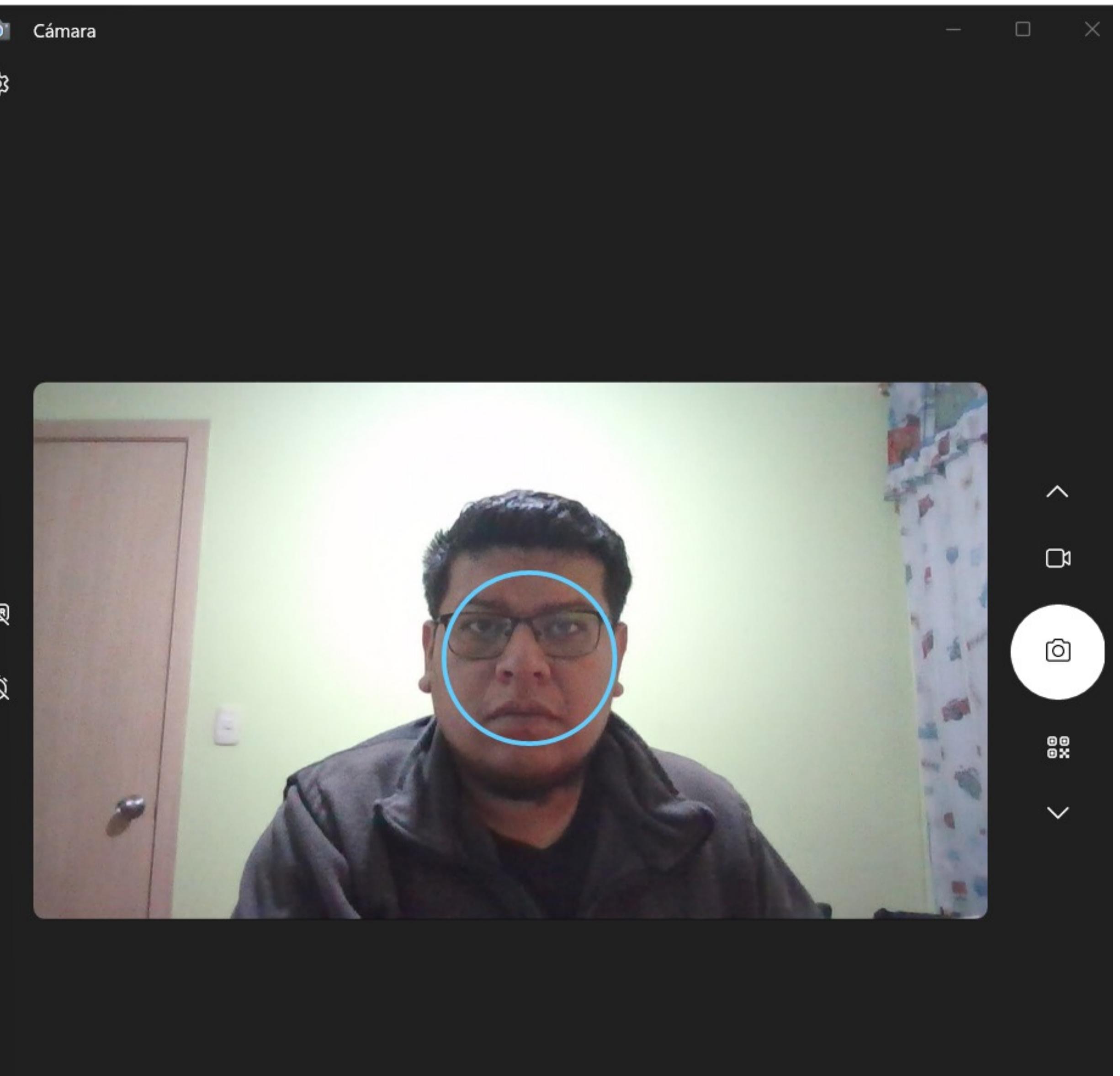
```
HERENCIA MULTIPLE.py X
HERENCIA MULTIPLE.py > ...
1  class Persona:
2      def __init__(self, nombre, edad, nacionalidad):
3          self.nombre = nombre
4          self.edad = edad
5          self.nacionalidad = nacionalidad
6
7      def hablar(self):
8          print(f"{self.nombre} esta hablando")
9
10 class Artista:
11     def __init__(self, habilidad):
12         self.habilidad = habilidad
13
14     def mostrar_habilidad(self):
15         return f"{self.habilidad}"
16
17 class EmpleadoArtista(Persona, Artista):
18     def __init__(self, nombre, edad, nacionalidad, habilidad, salario, empresa):
19         Persona.__init__(self, nombre, edad, nacionalidad)
20         Artista.__init__(self, habilidad)
21         self.salario = salario
22         self.empresa = empresa
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/HERENCIA MULTIPLE.py"

Hola soy: Fabricio, tengo la habilidad de Diseñar y trabajo en la empresa Picasso y mi sueldo es 2300

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>



Visual Studio Code

MRO.py

```
1 class A:
2     def hablar(self):
3         print("Hola desde A")
4
5 class F(A):
6     def hablar(self):
7         print("Hola desde F")
8
9
10 class B(A):
11     def hablar(self):
12         print("Hola desde B")
13
14 class C(F):
15     def hablar(self):
16         print("Hola desde C")
17
18 class D(B,C):
19     def hablar(self):
20         print("Hola desde D")
21
22 d = D()
23
24 F.hablar(A)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

Cámera

Hola desde F



Visual Studio Code

EJERCICIO 2.py

```
1 class Persona:
2     def __init__(self, nombre, edad):
3         self.nombre = nombre
4         self.edad = edad
5
6     def mostrar_datos(self):
7         print(f"El nombre de la persona es: {self.nombre}")
8         print(f"La edad de la persona es: {self.edad}")
9
10
11 class Estudiante(Persona):
12     def __init__(self, nombre, edad, semestre):
13         super().__init__(nombre, edad)
14         self.semestre = semestre
15
16     def mostrar_semestre(self):
17         print(f"El semestre del estudiante es: {self.semestre} semestre")
18
19 estudiante = Estudiante("Fabricio", "30 años", "Segundo")
20 estudiante.mostrar_datos()
21 estudiante.mostrar_semestre()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/EJERCICIO 2.py"
El nombre de la persona es: Fabricio
La edad de la persona es: 30 años
El semestre del estudiante es: Segundo semestre
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```

Cámara

EJERCICIO 2.py ...

El nombre de la persona es: Fabricio

La edad de la persona es: 30 años

El semestre del estudiante es: Segundo semestre

Camara

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Visual Studio Code
- Left Sidebar:** Icons for File, Find, Go, and others.
- Central Area:** A code editor displaying Python code. The code defines four classes: Animal, Mamifero, Herbivoro, and Zebra. The Zebra class inherits from both Mamifero and Herbivoro. Methods saltar, criar, and correr are defined in Animal, Mamifero, and Herbivoro respectively, and called on a Zebra instance.
- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, Python (with a plus sign), and other icons.

```
1  class Animal:
2      def saltar(self):
3          print("salta")
4
5
6  class Mamifero(Animal):
7      def criar(self):
8          print("cria")
9
10 class Herbivoro(Animal):
11     def correr (self):
12         print("corre")
13
14 class Zebra(Mamifero, Herbivoro):
15     pass
16
17 zebra2 = Zebra()
18
19 zebra2.saltar()
20 zebra2.criar()
21 zebra2.correr()
22
23
24
```

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/EJERCICIO 3.py"
```

salta

cria

corre

PS C:\U



Visual Studio Code

Cámara

POLIMORFISMO 1.py

```
1 class Animal():
2     def sonido(self):
3         pass
4 class Gato():
5     def sonido (self):
6         return "Miau"
7
8 class Perro():
9     def sonido(self):
10        return "guau"
11
12 def hacer_sonido(animal):
13     print(animal.sonido())
14
15 gato = Gato()
16 Perro= Perro()
17
18 hacer_sonido(gato)
19
20 print("Como hace el gato?", gato.sonido())
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Polimorfismo.py"
Miau
Como hace el gato? Miau
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```



Visual Studio Code

POLIMORFISMO 2.py X

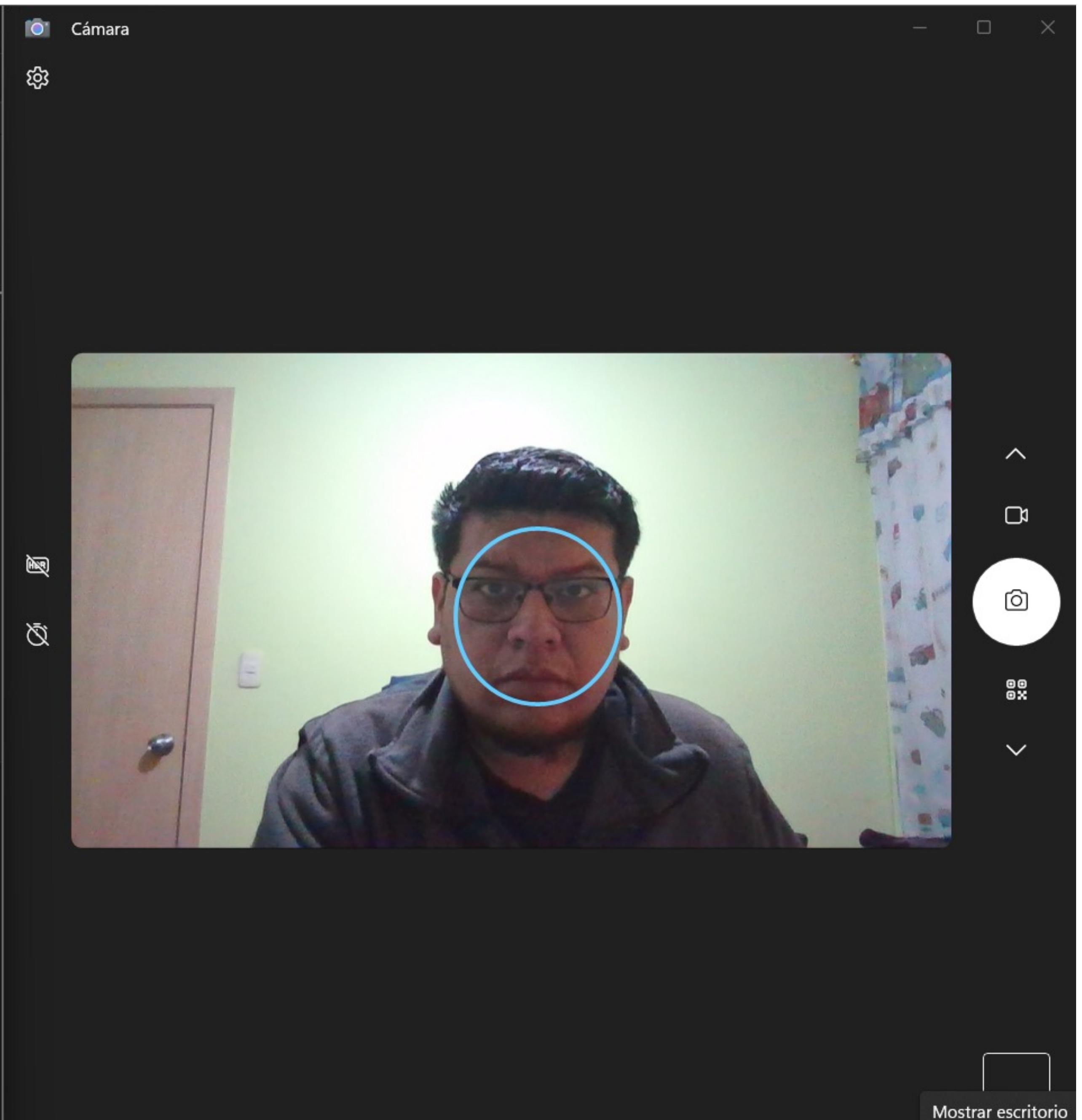
POLIMORFISMO 2.py > ...

```
1 def recorrer(elemento):
2     for item in elemento:
3         print(f"El elemento actual es: {item}")
4
5 lista1 = [1,2,3,4,5,6]
6 lista2 = ["HOLA","COMO","ESTAS?"]
7
8 recorrer(lista1)
9 recorrer(lista2)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + ▾

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/polimorfismo 2.py"
El elemento actual es: 1
El elemento actual es: 2
El elemento actual es: 3
El elemento actual es: 4
El elemento actual es: 5
El elemento actual es: 6
El elemento actual es: HOLA
El elemento actual es: COMO
El elemento actual es: ESTAS?
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```

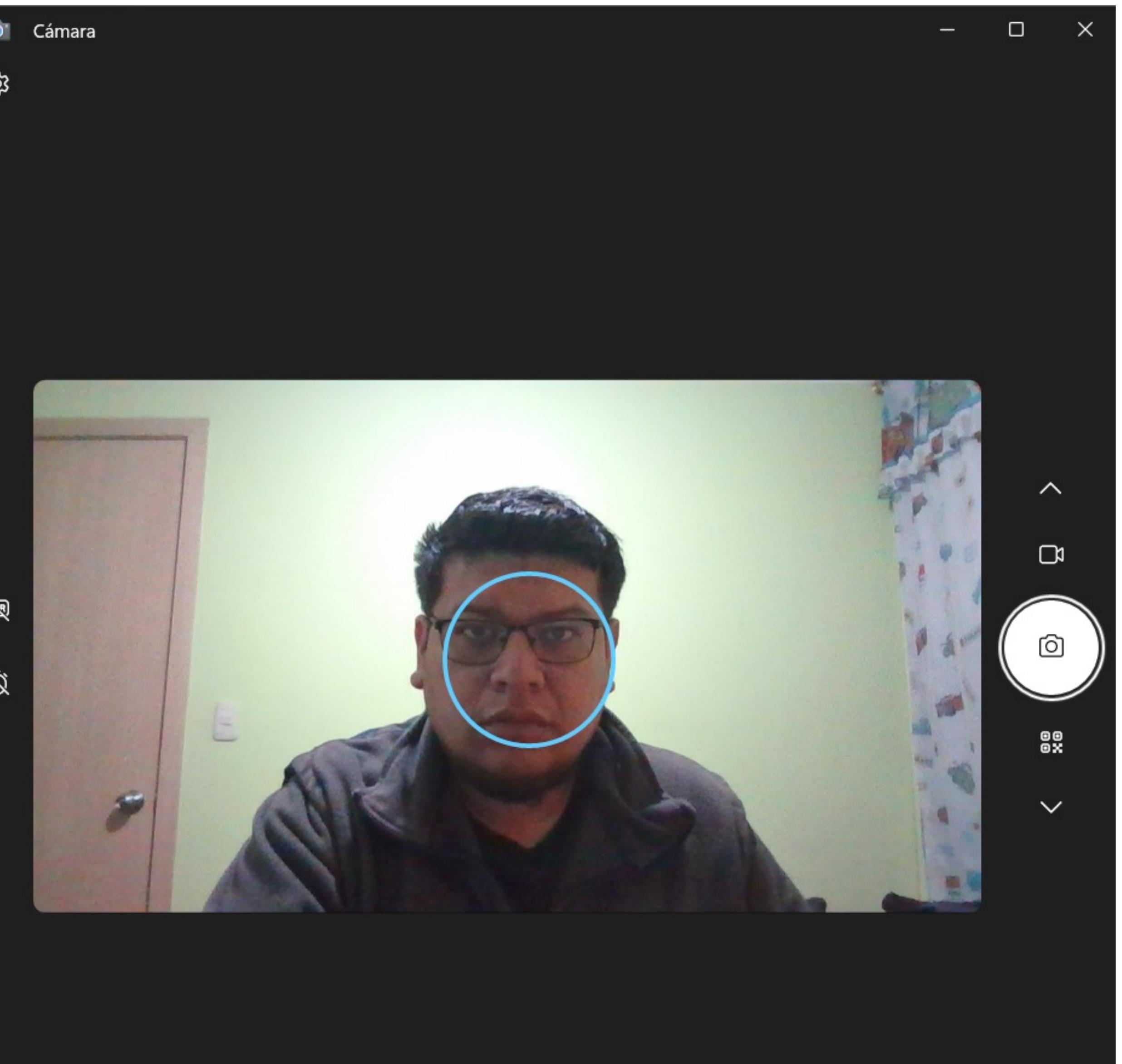


The screenshot shows the Visual Studio Code interface with a dark theme. On the left is the sidebar with icons for file operations like Open, Save, Find, and Refresh. The main area displays a Python script named 'ENCAPSULAMIENTO.py'. The code defines a class 'MiClase' with an __init__ method that initializes a private attribute '_atributo_privado' to 'valor'. It also contains a method '_hablar()' which prints a greeting. An object 'objeto' is created from the class, and both the private attribute and the method are printed. Below the code editor is the 'TERMINAL' tab, which shows the command-line output of running the script with Python 3.12.

```
class MiClase:
    def __init__(self):
        self._atributo_privado = "valor"
    def _hablar(self):
        print("Hola con quién tengo el gusto?")
objeto = MiClase()
print(objeto._atributo_privado)
print(objeto._hablar())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Encapsulamiento.py"
valor
Hola con quién tengo el gusto?
None
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>



Visual Studio Code

ENCAPSULAMIENTO FUERTE.py

```
1 class Clase:
2     def __init__(self):
3         self.__atributo_privado = "Valor"
4     def _hablar(self):
5         print("Hola estoy hablando con usted")
6 objeto2 = Clase()
7 print(objeto2._hablar())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Encapsulamiento Fuerte.py"

Hola estoy hablando con usted

None

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>

Cámara

A video feed from a camera showing a young man with dark hair and glasses, wearing a dark shirt. A blue circle highlights his face. The camera interface includes controls for zoom, crop, and settings.

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Visual Studio Code
- Left Sidebar:** Icons for File, Search, Open, Save, Find, and others.
- Active Editor:** METODOS GETTER Y SETTER.py
- Code Content:**

```
1 class Persona:
2     def __init__(self, nombre, edad):
3         self._nombre = nombre
4         self._edad = edad
5
6     def get_nombre(self):
7         return self._nombre
8
9     def set_nombre(self, new_nombre):
10        self._nombre = new_nombre
11
12 joven = Persona("Fernando", "31")
13
14 nombre = joven.get_nombre
15 print(joven._nombre)
16
17 joven.set_nombre("Fercho")
18 print(joven._nombre)
19
20 name = input("Ingrese el nombre de su personaje")
21 joven.set_nombre(name)
22 print(joven._nombre)
23
24
```
- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, Python (with a plus sign), and other icons.

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Metodos Getter y Setter.py"
```

Fernando

Fercho

Ingrrese el nombre de su personaje Fabricio

Fabrizio

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> █

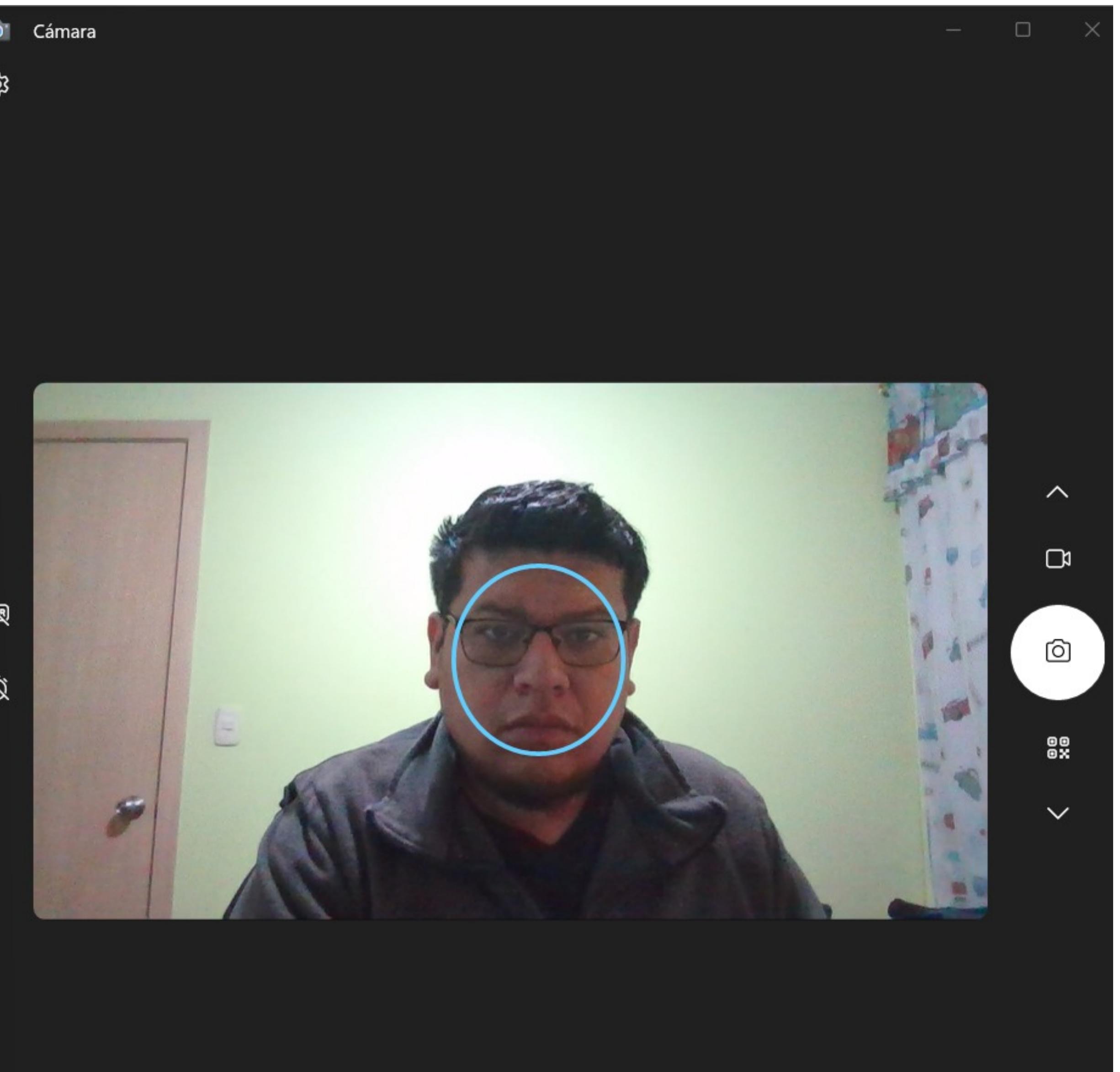


The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Visual Studio Code
- Left Sidebar:** Icons for File, Find, Go, Open, Save, and Terminal.
- Central Area:** A code editor window titled "METODOS GETTER Y SETTER 2.py". The code defines a Personaje class with __init__, get_nombre, and set_nombre methods, and demonstrates their use.

```
1 class Personaje:
2     def __init__(self, nombre, edad):
3         self.__nombre = nombre
4         self.__edad = edad
5
6     def get_nombre(self):
7         return self.__nombre
8
9     def set_nombre(self, new_nombre):
10        self.__nombre = new_nombre
11
12 Bakerlop = Personaje("Fernanda", 1)
13
14 nombre = Bakerlop.get_nombre()
15 print(nombre)
16
17 caracter = input("Ingrese el nombre del Personaje: ")
18
19 Bakerlop.set_nombre(caracter)
20
21 nombre = Bakerlop.get_nombre()
22 print("El nombre de tu personaje es ", nombre)
23
```

- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS, and a Python terminal tab.
- Terminal Output:** The terminal shows the execution of the script and its output: "Fernanda", followed by an input prompt "Ingrese el nombre del Personaje: Fabricio", the response "El nombre de tu personaje es Fabricio", and the final prompt "PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>".



Visual Studio Code

DECADOR.py

```
1 def decorador(funcion):
2     def funcion_modificada():
3         print("Antes de llamar a la función")
4         funcion()
5     return funcion_modificada
6
7 # def saludo():
8 #     print("Hola Fabricio")
9
10 # saludo_modificado = decorador(saludo)
11 # saludo_modificado()
12
13 @decorador
14 def saludo():
15     print("Hola Fabricio como estás?")
16
17 saludo()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

Cámara

Antes de llamar a la función
Hola Fabricio como estás?

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/DECADOR.py"

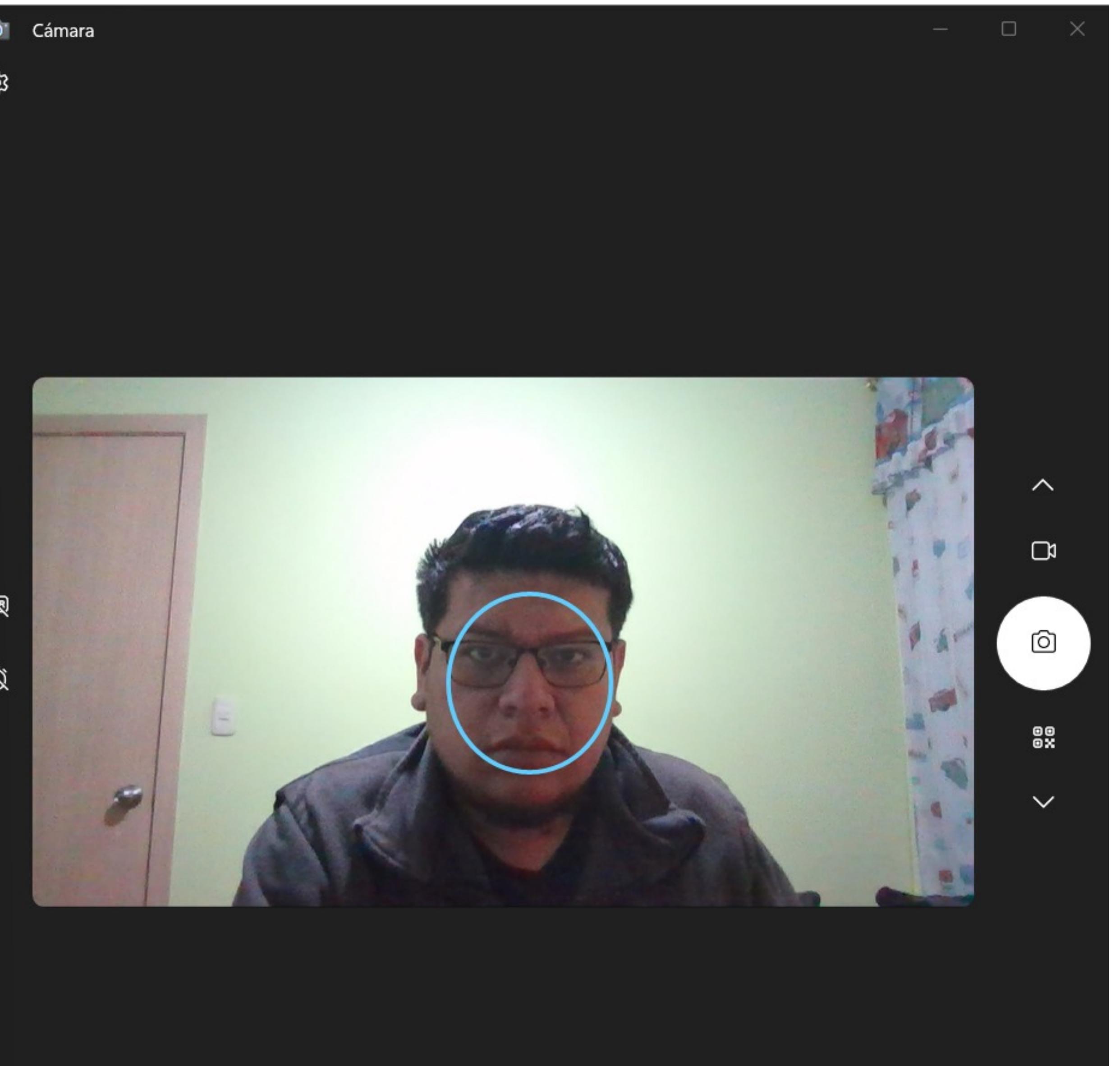


The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Visual Studio Code
- Left Sidebar:** Icons for File, Find, Go To, Open, Save, and Terminal.
- Central Area:** A code editor window titled "DECORADOR PROPERTY.py". The code implements a property decorator for a "Persona" class.

```
1  class Persona:
2      def __init__(self, nombre, edad):
3          self.__nombre = nombre
4          self.__edad = edad
5
6      @property
7      def nombre(self):
8          return self.__nombre
9
10     @nombre.setter
11     def nombre(self, new_nombre):
12         self.__nombre = new_nombre
13
14     @nombre.deleter
15     def nombre(self):
16         del self.__nombre
17
18     niño = Persona("Joaquín", "2")
19     nombre = niño.nombre
20     print(nombre)
21
22     niño.nombre = "Zack"
23     nombre = niño.nombre
24     print(nombre)
```
- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS.
- Terminal Tab:** Python
- Terminal Output:**

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Decorador Property.py"
Joaquín
Zack
Hello
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: "ABSTRACCIÓN.py" and "ABSTRACCIÓN.py > ...".
- Code Editor:** Displays the following Python code:

```
1 class Auto():
2     def __init__(self):
3         self._estado = "apagado"
4
5     def encender(self):
6         self._estado = "encendido"
7         print("El auto esta encendido")
8     def conducir(self):
9         if self._estado == "apagado":
10             self.encender()
11         print("Controlando el auto")
12
13 mi_auto = Auto()
14 mi_auto.conducir()
15
```
- Terminal:** Shows the command line output:

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Abstraccion.py"
El auto esta encendido
Controlando el auto
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```



Visual Studio Code

CLASES ABSTRACTAS.py

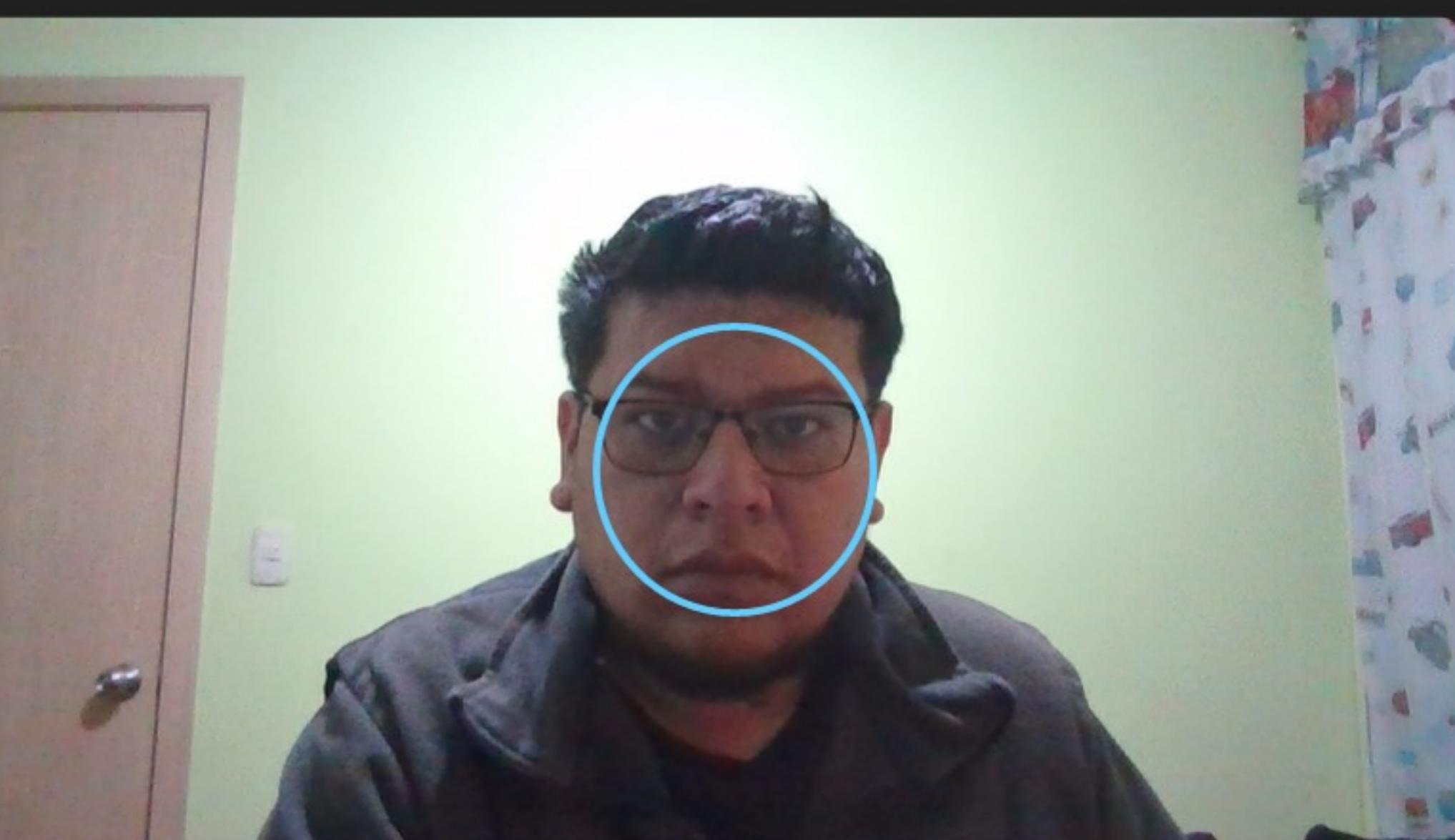
```
from abc import ABC, abstractclassmethod
class Persona(ABC):
    @abstractclassmethod
    def __init__(self, nombre, edad, sexo, actividad):
        self.nombre = nombre
        self.edad = edad
        self.sexo = sexo
        self.actividad = actividad
    @abstractclassmethod
    def hacer_actividad(self):
        pass
    def presentarse(self):
        print(f"Que tal mi nombre es {self.nombre} y tengo {self.edad} año
class Estudiante(Persona):
    def __init__(self, nombre, edad, sexo, actividad):
        super().__init__(nombre, edad, sexo, actividad)
    def hacer_actividad(self):
        print(f"Estoy estudiando: {self.actividad}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/CLASES ABSTRACTAS.py"
Que tal mi nombre es Fabricio y tengo 30 años de edad, soy de sexo masculino
Estoy estudiando: TICS
Que tal mi nombre es Fernando y tengo 30 años de edad, soy de sexo masculino
Soy futuro ingeniero
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```

Cámara



The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Visual Studio Code
- Left Sidebar:** Icons for File, Find, Replace, Go To, and others.
- Central Area:** A code editor window titled "METODOS ESPECIALES.py".
- Code Content:** Python code demonstrating special methods (`__init__`, `__str__`, `__repr__`, `__add__`) and their usage.

```
1 class Persona:
2     def __init__(self, nombre, edad):
3         self.nombre = nombre
4         self.edad = edad
5
6     def __str__(self):
7         return f"Persona(nombre={self.nombre}, edad ={self.edad})"
8
9     def __repr__(self):
10        return f'Persona("{self.nombre}", {self.edad})'
11
12    def __add__(self,otro):
13        nuevo_valor= self.edad + otro.edad
14        return Persona(self.nombre+otro.nombre,nuevo_valor)
15
16 Fabricio = Persona ("Fernando","30")
17 print(Fabricio)
18
19 lista = (1,2,3,4,5)
20 print(lista)
21
22 repre = repr(Fabricio)
23 resultado = eval(repre)
24 print(resultado.edad)
```

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/Metodos Especiales.py"
Persona(nombre=Fernando, edad =30)
(1, 2, 3, 4, 5)
30
Fernando
FernandoJavierVanessa
302029
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```



Visual Studio Code

EJERCICIO 3.py

```
1 class Personaje:
2     def __init__(self, nombre, fuerza, velocidad):
3         self.nombre = nombre
4         self.fuerza = fuerza
5         self.velocidad = velocidad
6
7     def __repr__():
8         return f"{self.nombre} (Fuerza: {self.fuerza}, Velocidad: {self.velocidad})"
9
10    def __add__(self, otro_pj):
11        nuevo_nombre = self.nombre + "-" + otro_pj.nombre
12        nueva_fuerza = round(((self.fuerza + otro_pj.fuerza)/2)**2)
13        nueva_velocidad = round(((self.velocidad + otro_pj.velocidad)/2)**2)
14        return Personaje(nuevo_nombre, nueva_fuerza, nueva_velocidad)
15
16 doraemon = Personaje("Doraemon",45,55)
17 gigante= Personaje("Gigante", 50,50)
18 novita = Personaje("Novita", 35,45)
19
20 dorami = doraemon + novita
21 gigantita = doraemon + gigante
22
23 print(doraemon)
24 print(gigante)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/EJERCICIO 3.py"

Doraemon (Fuerza: 45, Velocidad: 55)
Gigante (Fuerza: 50, Velocidad: 50)
Novita (Fuerza: 35, Velocidad: 45)
Doraemon-Novita (Fuerza: 1600, Velocidad: 2500)
Doraemon-Gigante (Fuerza: 2256, Velocidad: 2756)

Cámara



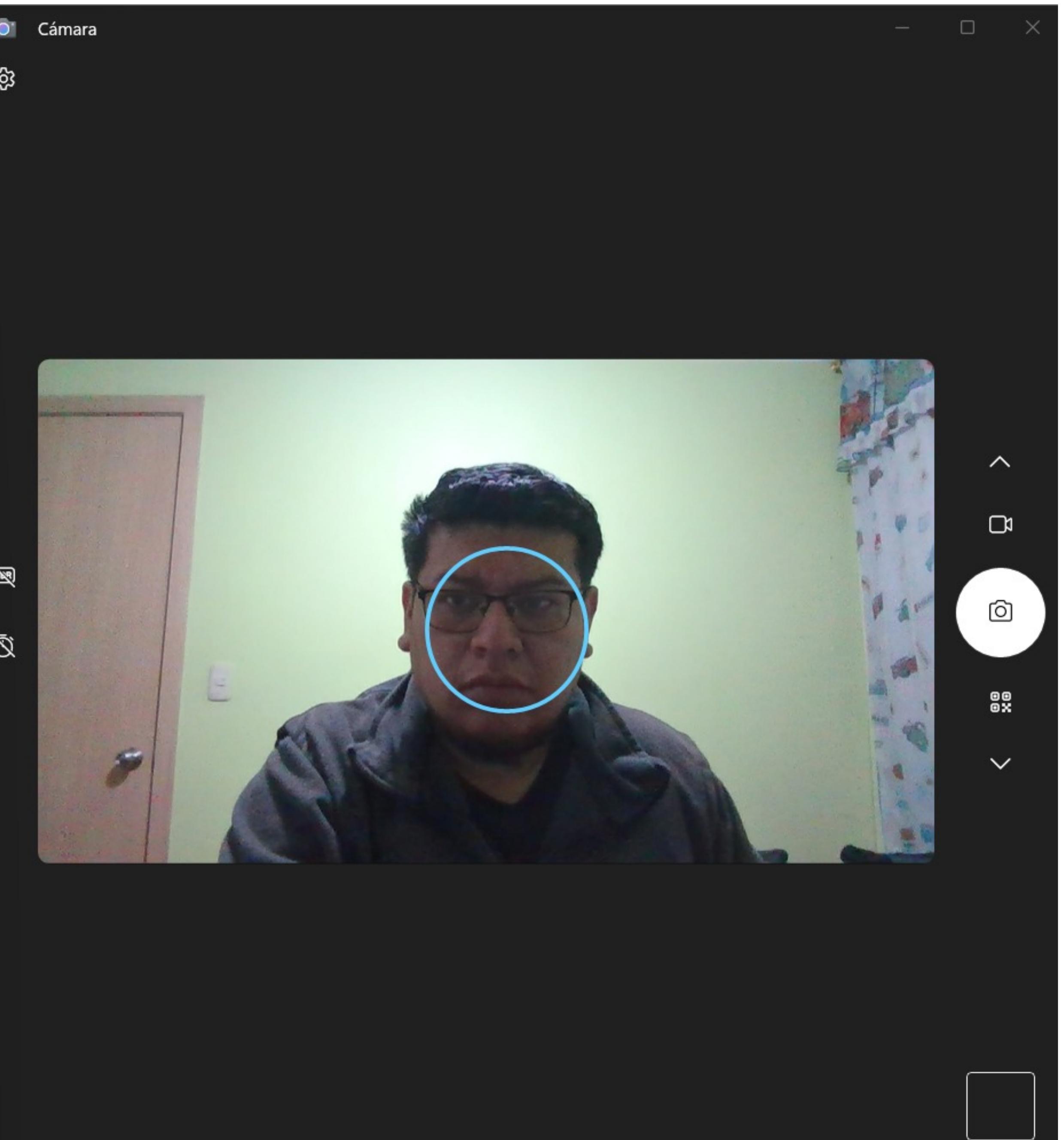
^ □ ☰

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a file named "SRP.py".
- Code Editor:** Displays the following Python code:

```
21     def agregar_combustible(self, cantidad):
22         self.combustible += cantidad
23
24     def obtener_combustible(self):
25         return self.combustible
26
27     def usar_combustible(self, cantidad):
28         self.combustible -= cantidad
29
30 tanque = TanqueDeCombustible()
31 autito = Auto(tanque)
32
33 print(autito.obtener_posición())
34 print(autito.mover(10))
35 print(autito.obtener_posición())
36 print(autito.mover(20))
37 print(autito.obtener_posición())
38 print(autito.mover(60))
39 print(autito.obtener_posición())
40 print(autito.mover(100))
41 print(autito.mover(100))
42 print(autito.obtener_posición())
```
- Terminal:** Shows the output of running the code with Python 3.12:

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/SRP.py"
0
Has movido el auto de manera satisfactoria
None
10
Has movido el auto de manera satisfactoria
None
30
Has movido el auto de manera satisfactoria
None
90
Has movido el auto de manera satisfactoria
```



The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Visual Studio Code
- Left Sidebar:** Icons for File, Search, Open, Save, Find, and others.
- Current File:** OCP.py
- Code Editor:** Displays the following Python code:

```
1 class Notificador:
2     def __init__(self, usuario, mensaje):
3         self.usuario = usuario
4         self.mensaje = mensaje
5
6     def notificar(self):
7         raise NotImplementedError
8
9 class NotificadorEmail(Notificador):
10    def Notificar(self):
11        print(f"Enviando mensaje por MAIL a {self.usuario.email}")
12
13 class NotificadorSMS(Notificador):
14    def Notificar(self):
15        print(f"Enviando SMS a {self.usuario.SMS}")
16
17 class NotificadorWhatsApp(Notificador):
18    def Notificar(self):
19        print(f"Enviando WhatsApp a {self.usuario.Whatsapp}")
20
21 class NotificadorTwitter(Notificador):
22    def Notificar(self):
23        print(f"Enviando Twitter a {self.usuario.Twitter}")
```

The code implements the Strategy design pattern, defining an abstract base class `Notificador` and four concrete subclasses: `NotificadorEmail`, `NotificadorSMS`, `NotificadorWhatsApp`, and `NotificadorTwitter`. Each subclass overrides the `notificar` method to implement a specific notification logic.

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/OCP.py"
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```



Visual Studio Code

LSP.py

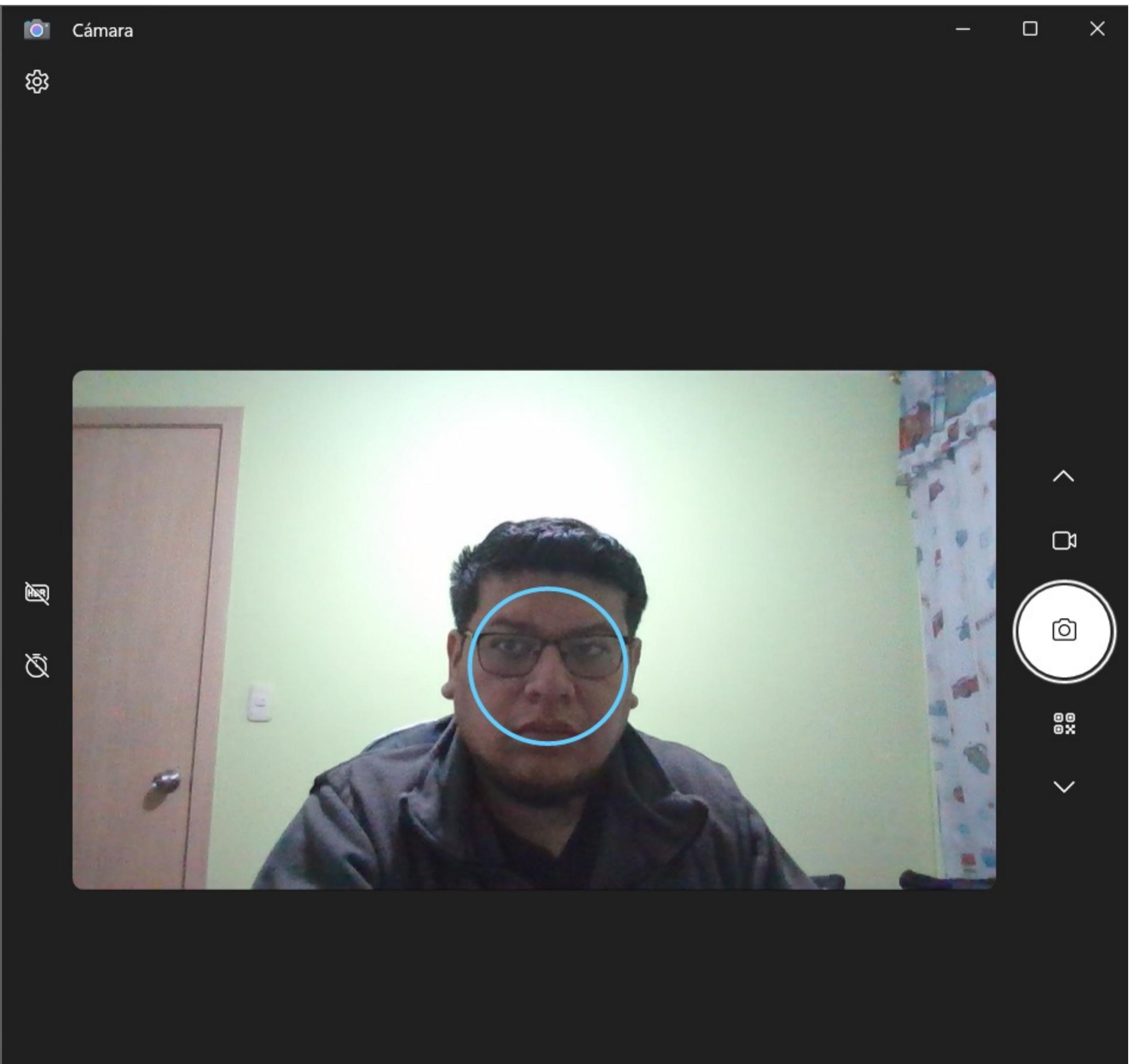
```
1 # class Ave:
2 #     def volar(self): return "Estoy volando"
3 #     class Pinguino(Ave):
4 #         def volar(self):
5 #             return "No puedo volar"
6
7 # def hacer_volar(ave = Ave):
8 #     return ave.volar()
9
10 # print(hacer_volar(Pinguino()))
11
12 class Ave:
13     pass
14
15 class AveVoladora(Ave):
16     def volar(self):
17         return "Estoy volando"
18
19 class AveNoVoladora(Ave):
20     pass
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

```
PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/LSP.py"

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>
```



Visual Studio Code

ISP.py

```
11     pass
12 class Humano(Trabajador):
13     def comer(self):
14         print("El humano está comiendo")
15
16     def trabajar(self):
17         print("El humano está trabajando")
18
19     def dormir(self):
20         print("El humano está durmiendo")
21
22 class Robot(Trabajador):
23     def comer(self):
24         pass
25
26     def trabajar(self):
27         print("El robot está trabajando")
28     def dormir(self):
29         pass
30 robot=Robot()
31 robot.trabajar()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code> & C:/Users/ingba/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ingba/OneDrive/Escritorio/Visual Studio Code/ISP.py"

El robot está trabajando

PS C:\Users\ingba\OneDrive\Escritorio\Visual Studio Code>

Cámara



