

■ TAREA HITO 2 ■

Programación de Sistemas Embebidos
UNIFRANZ
Sede El Alto
Hito2

JENSLLY CANAVIRI MAYDANA

¿Qué es un sistema embebido?

Un sistema embebido (también conocido como “empotrado”, “incrustado” o “integrado”) es un sistema de computación diseñado para realizar funciones específicas, y cuyos componentes se encuentran integrados en una placa base.

¿Mencione 5 ejemplos de sistemas embebidos?

- **Sistemas de calefacción central.**
- **Sistemas GPS.**
- **Rastreadores de fitness.**
- **Dispositivos médicos.**
- **Sistemas de automoción.**
- **Tránsito y cobro de tarifas.**
- **Cajeros automáticos.**
- **Robots de fábrica.**

¿Menciona las diferencias o similitudes entre un sistema operativo, un sistema móvil y un sistema embebido?

La diferencia entre un sistema operativo móvil (SO) y un sistema operativo de computadora tiene que ver con cómo las compañías tecnológicas individuales han implementado varias versiones de los sistemas operativos que proporcionan los entornos fundamentales para las aplicaciones de software tradicionales, así como las nuevas aplicaciones móviles.

¿A que se referirán los términos MCU y MPU? Explique cada una de ellas.

MCU = MCU son unas siglas que pueden hacer referencia a:

Microcontrolador, por sus siglas en inglés de microcontroller unit , un chip que contiene procesador, RAM, ROM, reloj y la unidad de control de E/S en un único encapsulado.

MPU = microchip procesador diseñado para realizar tareas múltiples dentro de un sistema de cómputo. Unidad de microprocesador (en inglés microprocessor unit) o unidad central de procesamiento.

¿Cuáles son los pilares de POO?

Los pilares son: abstracción, encapsulamiento, herencia y polimorfismo.

¿Mencione los componentes en lo que se basa POO?. Y explicar cada una de ellas.

El primer y más importante concepto de la POO es la distinción entre clase y objeto ya que la clase es una plantilla y esta se Define de manera genérica cómo van a ser los objetos de un determinado tipo.

Defina los siguientes:

Multiplataforma

Se denomina multiplataforma a un atributo conferido a programas informáticos o métodos y conceptos de cómputo que son implementados, y operan internamente en múltiples plataformas como en diferentes dispositivos

Multiparadigma

La Programación Multiparadigma es una práctica que emerge como resultado de los paradigmas orientado a objetos, procedural, declarativo y funcional.

Multipropósito

Una ontología para la representación del conocimiento difuso

Lenguaje interpretado

Un lenguaje interpretado es un lenguaje de programación para el que la mayoría de sus implementaciones ejecuta las instrucciones directamente.

Defina los siguientes:

- **Que es una Clase**

Una clase es la descripción de un conjunto de objetos similares; consta de métodos y de datos que resumen las características comunes de dicho conjunto.

- **Que es un Objeto**

Los objetos son instancias de clases. Ejemplo: Podríamos tener la clase Perro, una instancia de esta clase podría ser el objeto perro llamado "Chicho".

- **Que es una instancia**

Una instancia es un elemento tangible (ocupa memoria durante la ejecución del programa) generado a partir de una definición de clase. Todos los objetos empleados en un programa han de pertenecer a una clase determinada.

Defina a que se refiere cuando se habla de encapsulación y muestre un ejemplo(Código en Python).

```
class circulo():
    def __init__(self,radio):
        self.radio=radio

    @property
    def radio(self):
        print("Estoy dando el radio")
        return self.__radio

    @radio.setter
    def radio(self,radio):
        if radio>=0:
            self.__radio = radio
        else:
            print("Radio debe ser positivo")
            self.__radio=0
```

Defina a que se refiere cuando se habla de herencia y muestre un ejemplo(Código en Python)

```
# Definimos una clase padre
class Animal:
    pass

# Creamos una clase hija que hereda de la padre
class Perro(Animal):
    pass
```

De hecho podemos ver como efectivamente la clase `Perro` es la hija de `Animal` usando `__bases__`

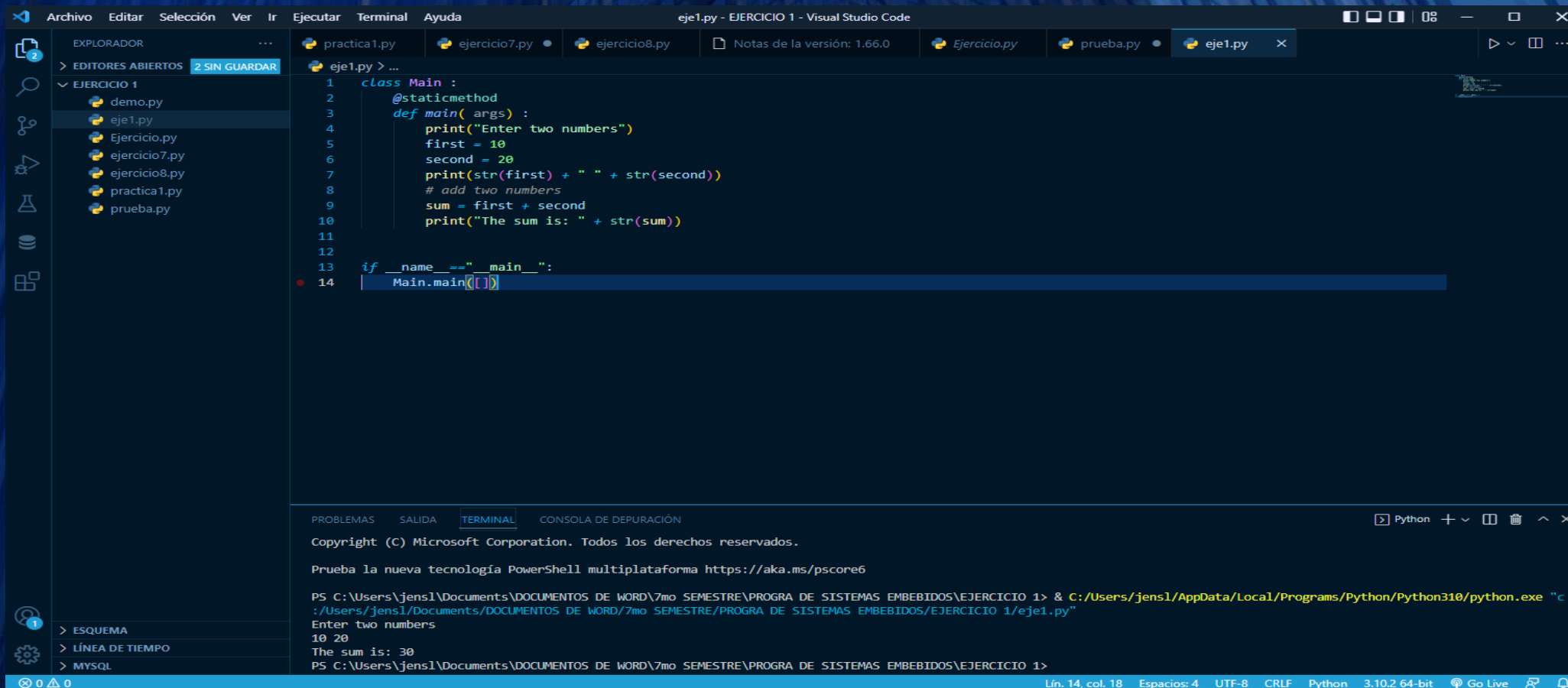
```
print(Perro.__bases__)
# (<class '__main__.Animal'>,,)
```

De manera similar podemos ver que clases descienden de una en concreto con `__subclasses__`.

```
print(Animal.__subclasses__())
# [<class '__main__.Perro'>]
```


PARTE PRACTICA

Llevar el siguiente código JAVA a Python.



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

- Explorer:** Shows a folder named 'EJERCICIO 1' containing files: 'demo.py', 'eje1.py', 'Ejercicio.py', 'ejercicio7.py', 'ejercicio8.py', 'practica1.py', and 'prueba.py'.
- Editor:** Displays the code for 'eje1.py'.
- Terminal:** Shows the execution of the script.

```
1 class Main :
2     @staticmethod
3     def main( args ) :
4         print("Enter two numbers")
5         first = 10
6         second = 20
7         print(str(first) + " " + str(second))
8         # add two numbers
9         sum = first + second
10        print("The sum is: " + str(sum))
11
12
13 if __name__ == "__main__":
14     Main.main()
```

Terminal Output:

```
PS C:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1> & C:\Users\jensl\AppData\Local\Programs\Python\Python310\python.exe "c:/Users/jensl/Documents/DOCUMENTOS DE WORD/7mo SEMESTRE/PROGRA DE SISTEMAS EMBEBIDOS/EJERCICIO 1/eje1.py"
Enter two numbers
10 20
The sum is: 30
PS C:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1>
```


Crear el código JAVA y Python para el siguiente análisis.



Propiedad
name
email
gender
nationality

Comportamiento
Write book
Write a movie
Change nationality
Change email

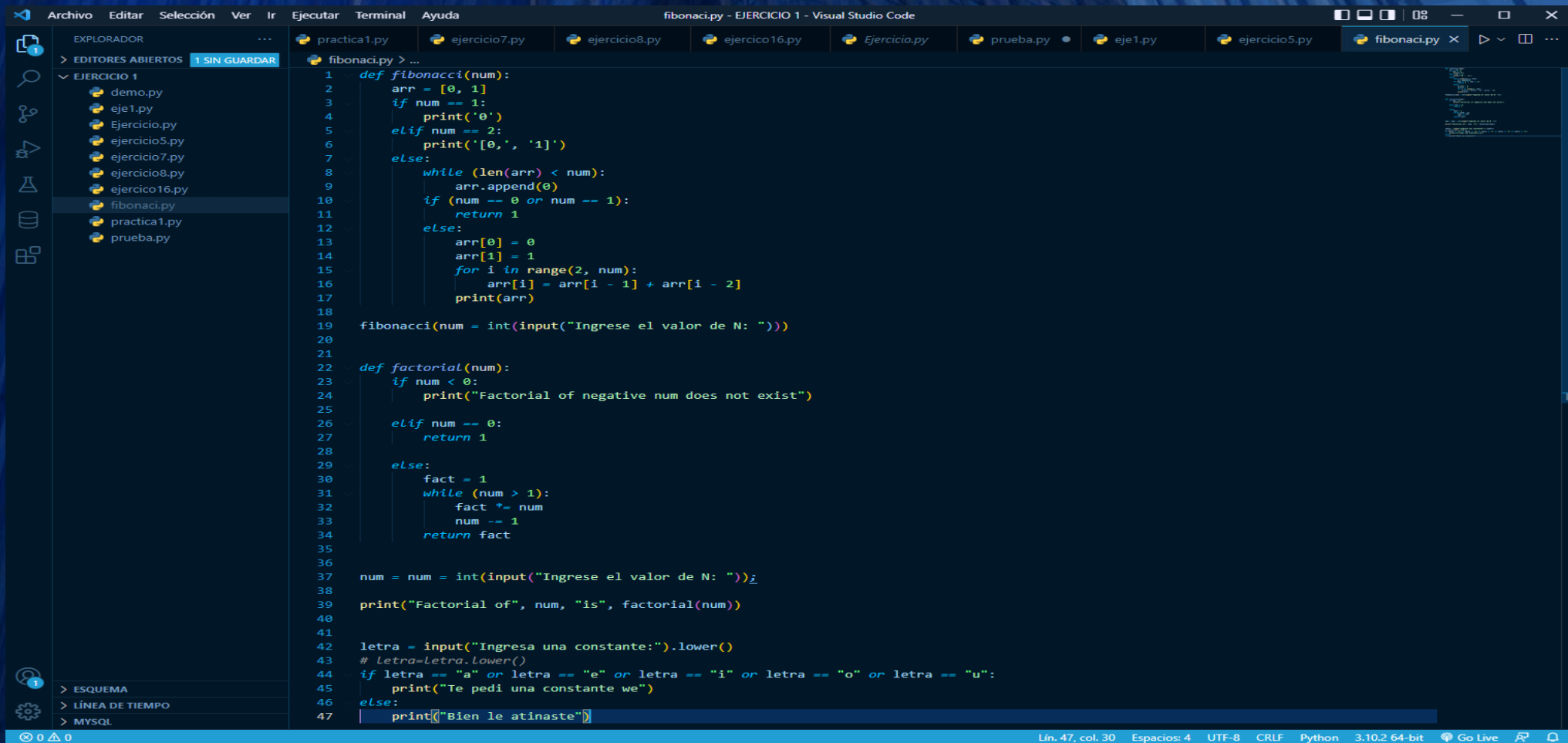
Visual Studio Code interface showing a Python file named `ejercicio5.py` with the following code:

```
1 class Libro:
2     nombre = None
3     email = None
4     genero = None
5     nacionalidad = None
6
7
8     def _init_(self, nombre, email, genero, nacionalidad):
9         self.nombre = nombre
10        self.email = email
11        self.genero = genero
12        self.nacionalidad = nacionalidad
13
14    def _str_(self):
15        return f'\nNombre: {self.nombre}, \nEmail: {self.email}, \nGenero: {self.genero}, \nNacionalidad: {self.nacionalidad}, \nLibro:'
16
17    def libro(self):
18        print("Mil y una noche")
19
20    def pelicula(self):
21        print("Guerra de las Galaxias")
22
23    def cambiarN(self):
24        print("Argentino")
25
26    def cambiarE(self):
27        print("gordon@gmail.com")
28
29    lec1 = Libro('Jenslly', 'jensllyc@gmail.com', 'Macho', 'Boliviano')
30    print(lec1)
31    Libro.libro
32    Libro.pelicula
33    Libro.cambiarN
34    Libro.cambiarE
```

The terminal output shows the execution of the script, resulting in a `TypeError: Libro() takes no arguments` error at line 29.

```
PS C:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1> & C:\Users\jensl\AppData\Local\Programs\Python\Python310\python.exe "c:/Users/jensl/Documents/DOCUMENTOS DE WORD/7mo SEMESTRE/PROGRA DE SISTEMAS EMBEBIDOS/EJERCICIO 1/ejercicio5.py"
Traceback (most recent call last):
  File "c:/Users/jensl/Documents/DOCUMENTOS DE WORD/7mo SEMESTRE/PROGRA DE SISTEMAS EMBEBIDOS/EJERCICIO 1/ejercicio5.py", line 29, in <module>
    lec1 = Libro('Jenslly', 'jensllyc@gmail.com', 'Macho', 'Boliviano')
TypeError: Libro() takes no arguments
PS C:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1>
```

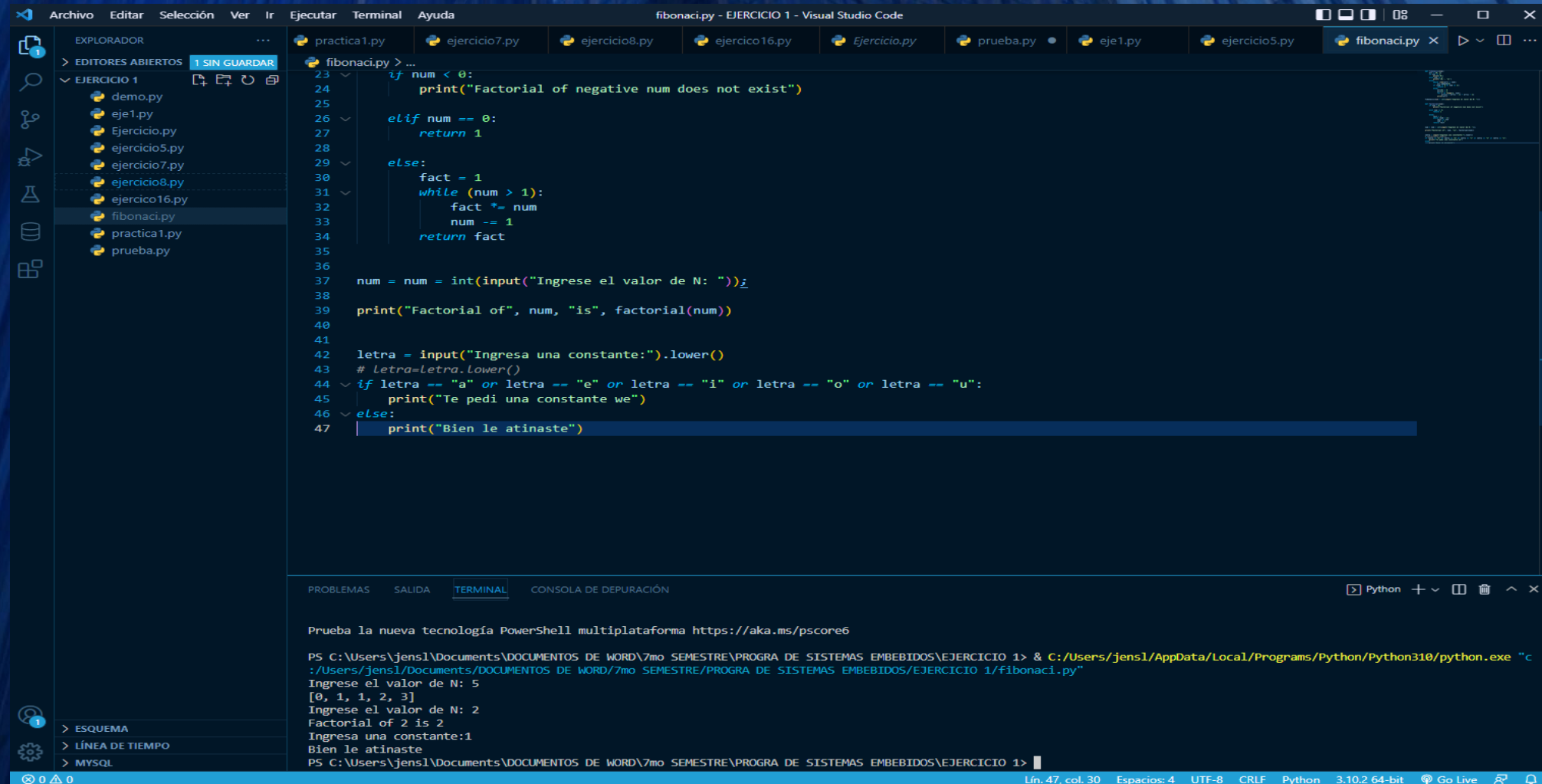

Crear un programa Python que genere los primeros N números de la serie Fibonacci.



The screenshot shows the Visual Studio Code interface with a Python file named `fibonaci.py` open. The code defines a `fibonacci` function that generates the first `num` Fibonacci numbers and stores them in an array. It also includes a `factorial` function and a main block that takes user input for `N` and prints the first `N` Fibonacci numbers.

```
1 def fibonacci(num):
2     arr = [0, 1]
3     if num == 1:
4         print('0')
5     elif num == 2:
6         print('[0, 1]')
7     else:
8         while (len(arr) < num):
9             arr.append(0)
10            if (num == 0 or num == 1):
11                return 1
12            else:
13                arr[0] = 0
14                arr[1] = 1
15                for i in range(2, num):
16                    arr[i] = arr[i - 1] + arr[i - 2]
17                print(arr)
18
19 fibonacci(num = int(input("Ingrese el valor de N: ")))
20
21
22 def factorial(num):
23     if num < 0:
24         print("Factorial of negative num does not exist")
25
26     elif num == 0:
27         return 1
28
29     else:
30         fact = 1
31         while (num > 1):
32             fact *= num
33             num -= 1
34         return fact
35
36
37 num = num = int(input("Ingrese el valor de N: "))
38
39 print("Factorial of", num, "is", factorial(num))
40
41
42 letra = input("Ingresa una constante:").lower()
43 # letra=letra.lower()
44 if letra == "a" or letra == "e" or letra == "i" or letra == "o" or letra == "u":
45     print("Te pedi una constante we")
46 else:
47     print("Bien le atinaste")
```


Crear un programa Python que genere los primeros N números de la serie Fibonacci.



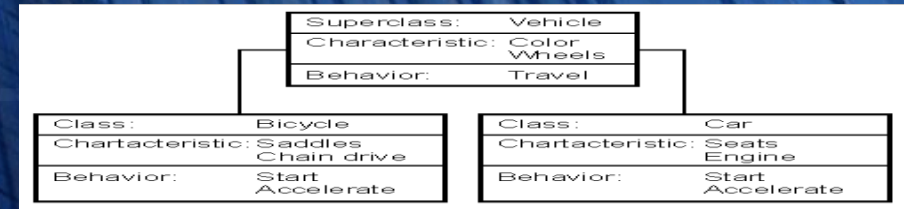
The image shows a screenshot of the Visual Studio Code editor with a Python file named `fibonaci.py` open. The file contains a program that calculates the factorial of a number and checks if a constant is a vowel. The terminal output shows the program being executed, with the user inputting 5 and 1, and the program outputting the factorial of 5 (120) and checking if 1 is a vowel (yes).

```
23 if num < 0:
24     print("Factorial of negative num does not exist")
25
26 elif num == 0:
27     return 1
28
29 else:
30     fact = 1
31     while (num > 1):
32         fact *= num
33         num -= 1
34     return fact
35
36
37 num = num = int(input("Ingrese el valor de N: "))
38
39 print("Factorial of", num, "is", factorial(num))
40
41
42 letra = input("Ingresa una constante:").lower()
43 # letra=letra.lower()
44 if letra == "a" or letra == "e" or letra == "i" or letra == "o" or letra == "u":
45     print("Te pedi una constante we")
46 else:
47     print("Bien le atinaste")
```

Terminal Output:

```
PS C:\Users\jens1\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1> & C:/Users/jens1/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/jens1/Documents/DOCUMENTOS DE WORD/7mo SEMESTRE/PROGRA DE SISTEMAS EMBEBIDOS/EJERCICIO 1/fibonaci.py"
Ingrese el valor de N: 5
[0, 1, 1, 2, 3]
Ingrese el valor de N: 2
Factorial of 2 is 2
Ingresa una constante:1
Bien le atinaste
PS C:\Users\jens1\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1>
```


Crear las clases necesarias para resolver el siguiente planteamiento.

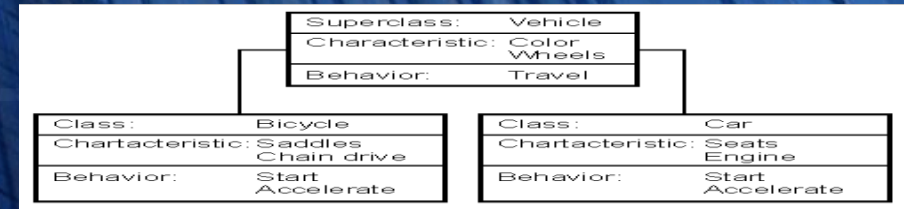


```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  ejercicio7.py - EJERCICIO 1 - Visual Studio Code

EXPLORADOR  ...
> EDITORES ABIERTOS  1 SIN GUARDAR
  EJERCICIO 1
    demo.py
    eje1.py
    Ejercicio.py
    ejercicio5.py
    ejercicio7.py
    ejercicio8.py
    ejercicio16.py
    fibonaci.py
    practica1.py
    prueba.py

ejercicio7.py > Car > _start_
1  class Vehiculo():
2      color = None
3      wheels = None
4      def __init__(self, color, wheels):
5          self.color = color
6          self.wheels = wheels
7
8      def _str_(self):
9          return "Vehiculo Color: {}, Cantidad ruedas  {}".format( self.color, self.wheels )
10
11
12  class Car(Vehiculo):
13
14      def __init__(self, color, wheels, seats, engine):
15          Vehiculo.__init__(self, color, wheels)
16          self.seats = seats
17          self.engine = engine
18
19      def _start_(self):
20          print("Encendiendo vehiculo")
21
22      def _accelerate_(self):
23          print("Acelerando vehiculo")
24
25      def _str_(self):
26          return Vehiculo._str_(self) + ", {} km/h, {} cc".format(self.seats, self.engine, self.start(), self.accelerate_())
27
28  c = Car("naranja", 7, 80, 2000)
29  print(c)
30  Car._start_
31  Car._accelerate_
32
33
34  class Bicycle(Vehiculo):
35
36      def __init__(self, color, wheels, saddles, chain):
37          Vehiculo.__init__(self, color, wheels)
38          self.saddles = saddles
39          self.chain = chain
40
41      def _startb_(self):
42          print("Iniciando Bici")
43
44      def _accelerateb_(self):
45          print("Acelerando bici")
46
47      def _str_(self):
48          return Vehiculo._str_(self) + ", {} sillones, {} cad " .format(self.saddles, self.chain, self._startb_(), self._accelerateb_())
49
```


Crear las clases necesarias para resolver el siguiente planteamiento.



```
46
47     def _str_(self):
48         return Vehiculo._str_(self) + ", {} sillas, {} cond.".format(self.saddles, self.chain, self.startb(), self.accelerateb_())
49
50 b = Bicycle("Rojo", 2, 4, 10)
51 print(b)
52 Bicycle._startb_
53 Bicycle._accelerateb_
```

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

```
PS C:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1> & C:/Users/jensl/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/jensl/Documents/DOCUMENTOS DE WORD/7mo SEMESTRE/PROGRA DE SISTEMAS EMBEBIDOS/EJERCICIO 1/ejercicio7.py"
Traceback (most recent call last):
  File "c:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1\ejercicio7.py", line 28, in <module>
    c = Car("naranja", 7, 80, 2000)
  File "c:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1\ejercicio7.py", line 15, in __init__
    Vehiculo._init_(self, color, wheels)
AttributeError: type object 'Vehiculo' has no attribute '_init_'. Did you mean: '__init__'?
PS C:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1>
```


Realizar un análisis para el siguiente escenario.



```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  ejercicio8.py - EJERCICIO 1 - Visual Studio Code
EXPLORADOR  1 SIN GUARDAR
> EDITORES ABIERTOS
  EJERCICIO 1
    demo.py
    eje1.py
    Ejercicio.py
    ejercicio5.py
    ejercicio7.py
    ejercicio8.py
    ejercicio16.py
    fibonacci.py
    practica1.py
    prueba.py
  ESQUEMA
  LÍNEA DE TIEMPO
  MYSQL

ejercicio8.py > ...
1  class poweredDevice:
2      nivel_de_potencia: None
3      estandares: None
4      control: None
5
6      def __init__(self, nivel_de_potencia, estandares, control):
7          self.nivel_de_potencia = nivel_de_potencia
8          self.estandares = estandares
9          self.Control = control
10
11      def __str__(self):
12          return f'nivel_de_potencia:{self.nivel_de_potencia}, \nestandares:{self.estandares}, \nControl:{self.Control}'
13
14      def set_edad(self, nueva_edad):
15          self.age = nueva_edad
16
17
18  power1 = poweredDevice('25W', '802.3at Tipo 2', 'PD')
19  print(power1)
20
21
22  class Scanner(poweredDevice):
23
24      tipo_s = None
25      calidad = None
26      marca = None
27
28      def __init__(self, nivel_de_potencia, estandares, control, tipo_s, calidad, marca):
29          poweredDevice.__init__(self, nivel_de_potencia, estandares, control)
30          self.tipo_s = tipo_s
31          self.calidad = calidad
32          self.marca = marca
33
34      def __str__(self):
35          return poweredDevice.__str__(self) + f' \nTipo_s:{self.tipo_s} \nCalidad:{self.calidad} \nMarca:{self.marca}'
36
37
38  est1 = Scanner('25W', '802.3at Tipo 2', 'PD', 'Mediano', 'Alta', 'Canon')
39  print(est1)
40
41
42  class Printer:
43
```


Realizar un análisis para el siguiente escenario.



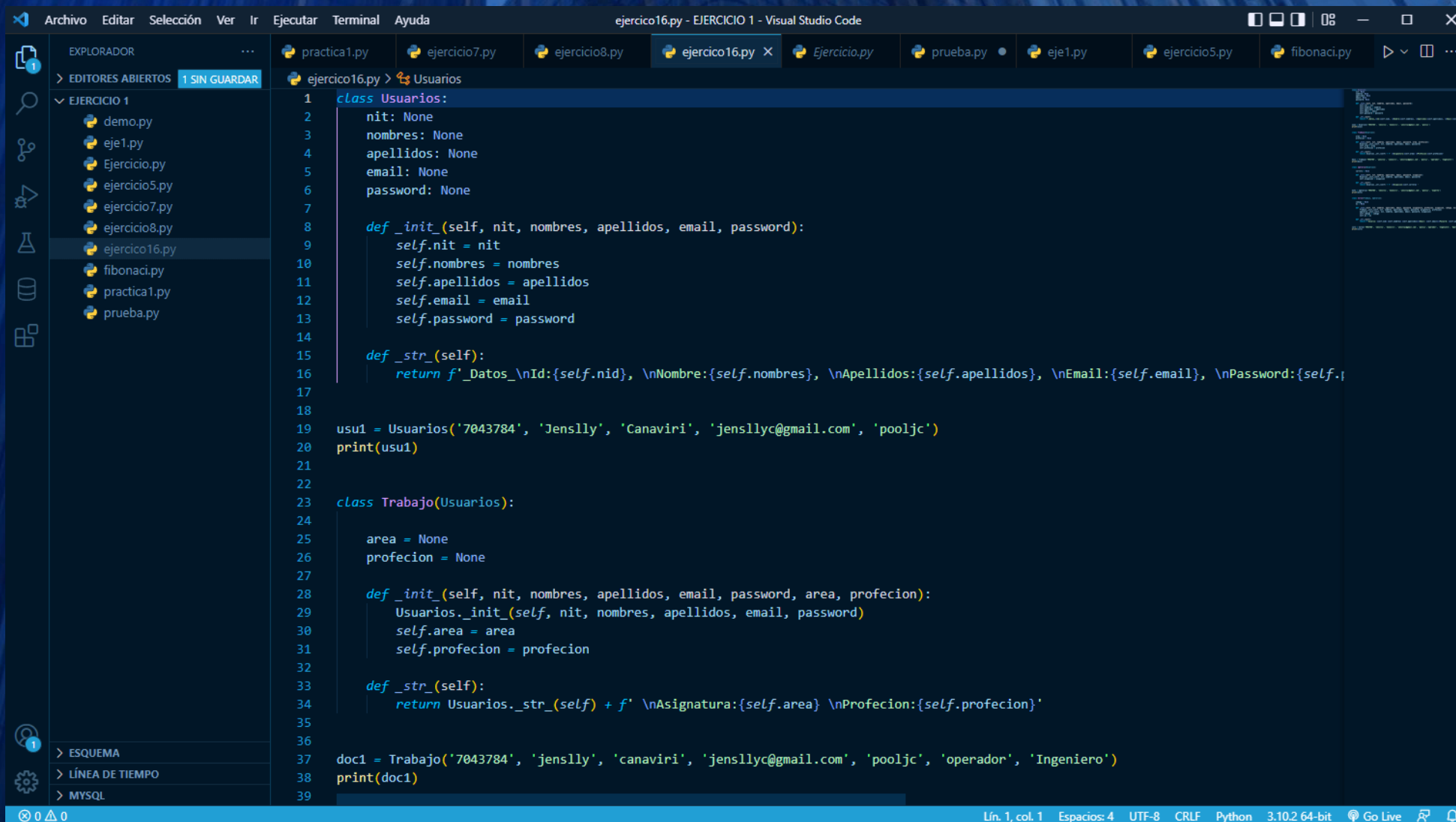
```
42 class Printer:
43
44     Capacidad_hojas = None
45     Modelo = None
46     Tipo_P = None
47
48     def __init__(self, nivel_de_potencia, estandares, control, Capacidad_hojas, Modelo, Tipo_P):
49         poweredDevice._init_(self, nivel_de_potencia, estandares, control)
50         self.Capacidad_hojas = Capacidad_hojas
51         self.Modelo = Modelo
52         self.Tipo_P = Tipo_P
53
54     def __str__(self):
55         return poweredDevice._str_(self) + f' \nCapacidad_hojas:{self.Capacidad_hojas} \nModelo:{self.Modelo} \nTipo_P:{self.Tipo_P}'
56
57
58 pri1 = Printer('25W', '802.3at Tipo 2', 'PD', 200, 'Canon', 'grande')
59 print(pri1)
60
61
62 class Copier:
63
64     Color_C = None
65     Marca_C = None
66     Calidad_C = None
67
68     def __init__(self, tipo_s, calidad, marca, Color_C, Marca_C, Calidad_C):
69         Scanner._init_(self, tipo_s, calidad, marca)
70         self.Color_C = Color_C
71         self.Marca_C = Marca_C
72         self.Calidad_C = Calidad_C
73
74     def __str__(self):
75         return Scanner._str_(self) + f' \nColor_C:{self.Color_C} \nMarca_C:{self.Marca_C} \nCalidad_C:{self.Calidad_C}'
76
77
78 cop1 = Copier('Mediano', 'Alta', 'Epson', 'Negro', 'epson', 'Baja')
79 print(cop1)
```

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

```
File "c:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1\ejercicio8.py", line 38, in <module>
    est1 = Scanner('25W', '802.3at Tipo 2', 'PD', 'Mediano', 'Alta', 'Canon')
File "c:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1\ejercicio8.py", line 29, in __init__
    poweredDevice._init_(self, nivel_de_potencia, estandares, control)
AttributeError: type object 'poweredDevice' has no attribute '_init_'. Did you mean: '__init__'?
PS C:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1>
```

Lin. 58, col. 58 Espacios: 4 UTF-8 CRLF Python 3.10.2 64-bit Go Live

Ejercicio de planteamiento



The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the output console on the right. The editor displays a Python file named `ejercicio16.py` with the following code:

```
1 class Usuarios:
2     nit: None
3     nombres: None
4     apellidos: None
5     email: None
6     password: None
7
8     def _init_(self, nit, nombres, apellidos, email, password):
9         self.nit = nit
10        self.nombres = nombres
11        self.apellidos = apellidos
12        self.email = email
13        self.password = password
14
15    def _str_(self):
16        return f'Datos_{self.nit}, {self.nombres}, {self.apellidos}, {self.email}, {self.password}'
17
18
19 usu1 = Usuarios('7043784', 'Jenslly', 'Canaviri', 'jensllyc@gmail.com', 'pooljc')
20 print(usu1)
21
22
23 class Trabajo(Usuarios):
24
25     area = None
26     profesion = None
27
28     def _init_(self, nit, nombres, apellidos, email, password, area, profesion):
29         Usuarios._init_(self, nit, nombres, apellidos, email, password)
30         self.area = area
31         self.profesion = profesion
32
33     def _str_(self):
34         return Usuarios._str_(self) + f'_{self.area}_{self.profesion}'
35
36
37 doc1 = Trabajo('7043784', 'jenslly', 'canaviri', 'jensllyc@gmail.com', 'pooljc', 'operador', 'Ingeniero')
38 print(doc1)
39
```

The output console on the right shows the output of the code:

```
Datos_7043784, Jenslly, Canaviri, jensllyc@gmail.com, pooljc
Datos_7043784, jenslly, canaviri, jensllyc@gmail.com, pooljc_operador_Ingeniero
```

The status bar at the bottom indicates the file is at line 1, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.10.2 64-bit, and the Go Live extension is active.

Ejercicio de planteamiento

The screenshot shows the Visual Studio Code interface with the file `ejercicio16.py` open. The code defines a `Usuarios` class and a `Curso` class that inherits from `Trabajo` and `operario`. The `Usuarios` class has attributes `carrera`, `nit`, `nombres`, `apellidos`, `email`, `password`, and `ocupacion`. The `Curso` class has attributes `codigo`, `ex`, `nit`, `nombres`, `apellidos`, `email`, `password`, `asignatura`, `profesion`, `ocupacion`, `codigo`, and `ex`. The code also includes initialization methods `__init__` and string representation methods `__str__` for both classes. The `Usuarios` class is instantiated as `est1` and the `Curso` class is instantiated as `cur1`.

```
40
41 class operario(Usuarios):
42
43     carrera = None
44
45     def __init__(self, nit, nombres, apellidos, email, password, ocupacion):
46         Usuarios.__init__(self, nit, nombres, apellidos, email, password)
47         self.ocupacion = ocupacion
48
49     def __str__(self):
50         return Usuarios.__str__(self) + f' \nOcupacion:{self.carrera} '
51
52
53 est1 = operario('7043784', 'Jenslly', 'Canaviri', 'jensllyc@gmail.com', 'pooljc', 'experto')
54 print(est1)
55
56
57 class Curso(Trabajo, operario):
58
59     codigo = None
60     ex = None
61
62     def __init__(self, nit, nombres, apellidos, email, password, asignatura, profesion, ocupacion, codigo, ex):
63         Trabajo.__init__(self, nit, nombres, apellidos, email, password, asignatura, profesion)
64         operario.__init__(self, nit, nombres, apellidos, email, password, ocupacion)
65         self.codigo = codigo
66         self.ex = ex
67
68     def __str__(self):
69         return f'Usuario: {self.nid} {self.nombres} {self.apellidos}\nEmail: {self.email}\nPasword: {self.password}\nArea: {self.asigna
70
71
72 cur1 = Curso('7043784', 'Jenslly', 'Canaviri', 'jensllyc@gmail.com', 'pooljc','operador', 'Ingeniero', 'Operario', 777, '5 años')
73 print(cur1)
```

The terminal window shows a traceback error:

```
Traceback (most recent call last):
  File "c:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1\ejercicio16.py", line 19, in <module>
    usu1 = Usuarios('7043784', 'Jenslly', 'Canaviri', 'jensllyc@gmail.com', 'pooljc')
TypeError: Usuarios() takes no arguments
PS C:\Users\jensl\Documents\DOCUMENTOS DE WORD\7mo SEMESTRE\PROGRA DE SISTEMAS EMBEBIDOS\EJERCICIO 1>
```