EXERCISES QUE TRABAJAREMOS EN EL CUE

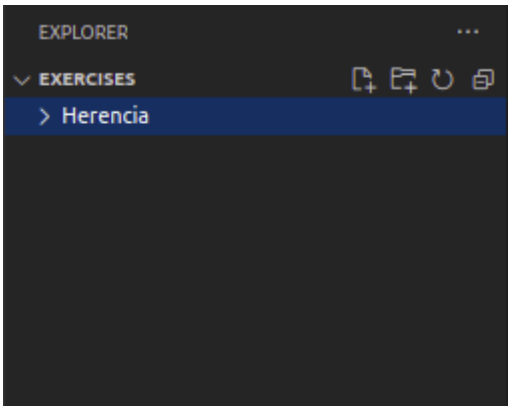
- EXERCISE 1: CREACIÓN DE UNA CLASE.
- EXERCISE 2: HERENCIA SIMPLE.
- EXERCISE 3: POLIMORFISMO.

EXERCISE 1: CREACIÓN DE UNA CLASE

Construiremos una clase Persona, que contiene los siguientes atributos y métodos:

- Atributos: cedula, nombre y apellidos.
- Métodos: imprimir_persona

Abrimos Visual Studio Code (VSC), y procedemos a crear en el mismo, una carpeta llamada herencia:



Procedemos a crear dos archivos, uno principal llamado `main.py` donde realizaremos el llamado de la clase creada, y otro llamado `persona.py` donde se definen las clases.

En el archivo `persona.py` procedemos a crear la clase:

Herencia/persona.py

```
class Persona:
    def __init__(self, nombre, apellidos, cedula):
        self.nombre = nombre
        self.apellidos = apellidos
        self.cedula = cedula
```

Seguidamente, procedemos a crear el método imprimir datos:

Herencia/persona.py

```
class Persona:
    def __init__(self, nombre, apellidos, cedula):
        self.nombre = nombre
        self.apellidos = apellidos
        self.cedula = cedula

    def imprimir_datos(self):
        print( f'Nombre: {self.nombre} \nApellidos: {self.apellidos}
\nCédula: {self.cedula}')
```

Finalmente, procedemos a crear el método `__str__` que devuelve la representación del objeto Persona como una cadena:

Herencia/persona.py

```
class Persona:
    def __init__(self, nombre, apellidos, cedula):
        self.nombre = nombre
        self.apellidos = apellidos
        self.cedula = cedula

    def imprimir_datos(self):
        print( f'Nombre: {self.nombre} \nApellidos: {self.apellidos}
\nCédula: {self.cedula}')

    def __str__(self):
        return f'Nombre: {self.nombre} \nApellidos: {self.apellidos}
\nCédula: {self.cedula}
```

Creamos los objetos persona en archivo `main.py`.

En primer lugar, en este archivo debemos importar la clase Persona.

Herencia/main.py

```
from persona import Persona
```

Seguidamente, procedemos a crear los objetos del tipo persona:

Herencia/main.py

```
from persona import Persona

persona1 = Persona('Juan', 'Perez','123456')
persona2 = Persona('José', 'Sanchez','456789')
```

Procedemos a hacer uso de llamado de los métodos de `imprimir_datos`, y de la devolución del del objeto persona:

Herencia/main.py

```
from persona import Persona

persona1 = Persona('Juan', 'Perez','123456')
persona2 = Persona('José', 'Sanchez','456789')

persona1.imprimir_datos()
print(persona1)

persona2.imprimir_datos()
print(persona2)
```

Finalmente, en la terminal tenemos al ejecutar el archivo `main.py`

Terminal

```
$ PYTHON MAIN.PY
NOMBRE: JUAN
APELLIDOS: PEREZ
CÉDULA: 123456
NOMBRE: JUAN
APELLIDOS: PEREZ
CÉDULA: 123456
NOMBRE: JOSÉ
APELLIDOS: SANCHEZ
CÉDULA: 456789
NOMBRE: JOSÉ
APELLIDOS: SANCHEZ
CÉDULA: 456789
```

EXERCISE 2: HERENCIA SIMPLE

Comenzamos a definir dos nuevas clases que heredan los atributos y los métodos de la clase Persona, esto son los tipos de objeto o clases supervisor y cliente, respectivamente. El supervisor contiene un atributo llamado `zona`, y un nuevo método de `imprimir_empleado` con los datos del empleado; y el cliente contiene un nuevo atributo llamado `porcentaje de descuento`, y un nuevo método de `imprimir_cliente` con los datos del Cliente.

Procedemos a adecuar la respectiva clase Persona agregando la clase Supervisor:

Herencia/persona.py

```
.....
class Supervisor(Persona):
    def __init__(self, nombre, apellidos, cedula, zona):
        super().__init__(nombre, apellidos, cedula)
        self.zona = zona
```

Adecuamos el método de **imprimir_supervisor**:

Herencia/personpy

```
.....
def imprimir_supervisor(self):
        super().imprimir_persona()
        print('Zona:',self.zona)
```

Finalmente, reescribimos el método **__str__**

Herencia/persona.py

```
.....
def __str__(self):
        return super().__str__() + f'\nZona: {self.zona}'
```

Procedemos con la construcción de la clase Cliente:

Herencia/persona.py

```
.....
class Cliente(Persona):
    def __init__(self, nombre, apellidos, cedula, descuento ):
        super().__init__(nombre, apellidos, cedula)
        self.descuento = descuento
```

Agregamos el método **imprimir_cliente**:

Herencia/persona.py

```
.....
def imprimir_cliente(self):
        super().imprimir_persona()
        print('Descuento: ', self.descuento)
```

Finalmente reescribimos el método **__str__** para la clase Cliente:

Herencia/persona.py

```
.....
def __str__(self):
        return super().__str__() + f'\nDescuento: {self.descuento}'
```

Adecuamos el archivo principal para crear y procesar cada uno de los objetos. Procedemos a construir un objeto Supervisor:

Herencia/persona.py

```
from persona import Supervisor

supervisor1 = Supervisor('Juan', 'Perez','123456', 'Sur')
print("******")
supervisor1.imprimir_persona()
print("******")
supervisor1.imprimir_supervisor()
print("******")
print(supervisor1)
```

Terminal

```
$ python main.py
******
******
Nombre: Juan
Apellidos: Perez
Cédula: 123456
******
Nombre: Juan
Apellidos: Perez
Cédula: 123456
Zona: Sur
******
Nombre: Juan
Apellidos: Perez
Cédula: 123456
Zona: Sur
```

Procedemos a construir un objeto Cliente:

Herencia/persona.py

```
from persona import Supervisor, Cliente

cliente1 = Cliente('Juan', 'Perez','123456', 20)
print("******")
```

```
cliente1.imprimir_persona()
print("******")
cliente1.imprimir_cliente()
print("******")
print(cliente1)
```

Terminal

```
$ python main.py
******
Nombre: Juan
Apellidos: Perez
Cédula: 123456
******
Nombre: Juan
Apellidos: Perez
Cédula: 123456
Descuento:  20
******
Nombre: Juan
Apellidos: Perez
Cédula: 123456
Descuento: 20
```

EXERCISE 3: POLIMORFISMO

En este caso vamos a definir un nuevo método en la clase persona que es `get_tipo()`, y que imprima que “Soy del tipo persona”, “Soy del tipo supervisor”, y “Soy del tipo cliente”.

Procedemos a adecuar la clase:

Herencia/persona.py

```
class Persona:
    def __init__(self, nombre, apellidos, cedula):
        self.nombre = nombre
        self.apellidos = apellidos
        self.cedula = cedula

    def get_tipo(self):
        print("Soy del tipo Persona")

    def imprimir_persona(self):
        print( f'Nombre: {self.nombre} \nApellidos: {self.apellidos}
\nCédula: {self.cedula}')

    def __str__(self):
```

```
        return f'Nombre: {self.nombre} \nApellidos: {self.apellidos}
\nCédula: {self.cedula}'
```

Herencia/main.py

```
from persona import Persona, Supervisor, Cliente

print("******")
persona1 = Persona('Juan', 'Perez','123456')
persona1.get_tipo()

print("******")
supervisor1 = Supervisor('Juan', 'Perez','123456', 'Sur')
supervisor1.get_tipo()

print("******")
cliente1 = Cliente('Juan', 'Perez','123456', 20)
cliente1.get_tipo()
```

La salida:

Terminal

```
$ python main.py
******
Soy del tipo Persona
******
Soy del tipo Persona
******
Soy del tipo Persona
```

Adecuando y aplicando polimorfismo, con la finalidad de sobrescribir cada uno de los métodos de cada una de las subclases y que al llamar el método `get_tipo()`, imprima: "Soy del tipo persona", "Soy del tipo supervisor", y "Soy del tipo cliente", según el objeto **Persona**, **Supervisor** y **Cliente** respectivamente:

Adecuamos el código de la siguiente manera:

Herencia/persona.py

```
class Persona:
    def __init__(self, nombre, apellidos, cedula):
        self.nombre = nombre
        self.apellidos = apellidos
        self.cedula = cedula

```

```
    def get_tipo(self):
        print("Soy del tipo Persona")

    def imprimir_persona(self):
        print( f'Nombre: {self.nombre} \nApellidos: {self.apellidos}
\nCédula: {self.cedula}')

    def __str__(self):
        return f'Nombre: {self.nombre} \nApellidos: {self.apellidos}
\nCédula: {self.cedula}'

class Supervisor(Persona):
    def __init__(self, nombre, apellidos, cedula, zona):
        # super().__init__(nombre, apellidos, cedula)
        Persona.__init__(self, nombre, apellidos, cedula)
        self.zona = zona

    def get_tipo(self):
        print("Soy del tipo Supervisor")

    def imprimir_supervisor(self):
        # super().imprimir_persona()
        Persona.imprimir_persona(self)
        print('Zona:',self.zona)

    def __str__(self):
        # return super().__str__() + f'\nZona: {self.zona}'
        return Persona.__str__(self) + f'\nZona: {self.zona}'

class Cliente(Persona):
    def __init__(self, nombre, apellidos, cedula, descuento ):
        super().__init__(nombre, apellidos, cedula)
        self.descuento = descuento

    def get_tipo(self):
        print("Soy del tipo Cliente")

    def imprimir_cliente(self):
        super().imprimir_persona()
        print('Descuento: ', self.descuento)

    def __str__(self):
        return super().__str__() + f'\nDescuento: {self.descuento}'
```

La salida:

Terminal

```
$ python main.py
******
Soy del tipo Persona
```

```
******
Soy del tipo Supervisor
******
Soy del tipo Cliente
```