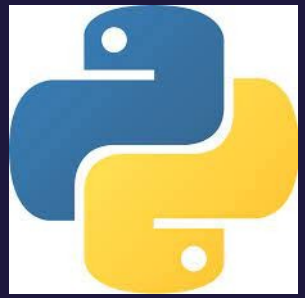
LENGUAJES y HERRAMIENTA PARA CIENCIAS DE DATOS I

POO en Python III
(Herencia y Polimorfismo)



UCO ONLINE

Herencia

- Reutilizar un clase extendiendo su funcionalidad

```
class CuentaAhorro(CuentaBancaria):
    interes = 5

    def __init__(self, cuenta, titular, limite):
        super().__init__(cuenta, titular)
        self.limite = limite

    def pagar_recibo(self, concepto, importe):
        self.saldo = self.saldo - importe
        print(f'Se ha pagado un recibo de {importe} correspondiente a {concepto}')
```

```
a = CuentaAhorro(1024, 'David', 1000)
a.ingresar(120)
a.pagar_recibo('Ingles', 120)
```

Herencia

- `type(variable)`
  - Tipo de un variable u objeto
- `isinstance(objeto, Clase)`
  - Determina si un objeto es una instancia de una clase o de su superclase
- issubclass(ClaseA, ClaseB)
  - Determina si la ClaseA hereda de la ClaseB

Herencia múltiple

- Heredar de dos padres

```
class Coche():
    def print_coche (self):
        print('Tierra')

class Barco():
    def print_barco(self):
        print('Agua')

class Amfibio(Coche,Barco):
    def print_amfibio(self):
        self.print_coche()
        self.print_barco()

a = Amfibio()

a.print_amfibio()
```

Encapsulación

- Capacidad de un objeto para ocultar su estado
  - Acceder mediante los métodos definidos
- Python
  - Todo el público
  - Usar el _ para indicar que no se debería modificar
  - Usando __ el atributo es totalmente oculto

Polimorfismo

- Invocar un mismo método sobre diferentes objetos y obtener diferentes resultados

```
c = Circulo(2)
t = Triangulo (2, 3)
r = Rectangulo (4, 5)

figuras = (c, t, r)

for i in figuras:
    print(i.area())
```