Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Docente:

Ing. Coyla Idme Leonel

Alumno:

Ticona Miramira Roberto Angel

Herencia en POO

La herencia es un concepto fundamental en la POO que permite crear nuevas clases a partir de clases existentes. La clase nueva llamada clase derivada o subclase hereda atributos y métodos de la clase original (llamada clase base) o superclase. Esto facilita la reutilización de código y la organización jerarquica de las clases.

» Herencia simple

Una clase derivada hereda de una unida clase base, esto significa que la subclase solo tiene una superclase directa. Es el tipo de herencia más común y sencillo de implementar.

Ejemplos

1.- Crear la clase animal y heredar en las clases perro y gato.

» Código

```python
class Animal: # clase base
    def __init__(self, nombre):
        self.nombre = nombre

    def hacerSonido(self):
        pass

class Perro(Animal): # clase derivada
    def hacerSonido(self):
        return "¡Guau!"

class Gato(Animal):
    def hacerSonido(self):
        return "¡Miauuu!"

perro = Perro("Rex")
print(f"{perro.nombre} dice {perro.hacerSonido()}")

gato = Gato("Charlotte")
print(f"{gato.nombre} dice {gato.hacerSonido()}")
```

» Ejecución

```
Rex dice ¡Guau!
Charlotte dice ¡Miauuu!
```

2.- Crear la clase FiguraGeometrica y luego crear de ella las clases Circulo y Rectangulo, calculando sus áreas y perímetros.

» Código

```python
import math
class FiguraGeometrica:
    def __init__(self, nombre):
        self.nombre = nombre

    def area(self):
        raise NotImplementedError("Subclases deben implementar este método")

    def perimetro(self):
        raise NotImplementedError("Subclases deben implementar este método")

class Circulo(FiguraGeometrica):
    def __init__(self, radio):
        super().__init__("Circulo")
        self.radio = radio

    def area(self):
        return math.pi * (self.radio ** 2)

    def perimetro(self):
        return 2 * math.pi * self.radio

class Rectangulo(FiguraGeometrica):
    def __init__(self, base, altura):
        super().__init__("Rectángulo")
        self.base = base
        self.altura = altura


    def area(self):
        return self.base * self.altura

    def perimetro(self):
        return 2 * (self.base + self.altura)

circulo = Circulo(5)
print(f"Nombre: {circulo.nombre}")
print(f"Área: {circulo.area():.2f}")
print(f"Perímetro: {circulo.perimetro():.2f}")
rectangulo = Rectangulo(8,6)
print(f"Nombre: {rectangulo.nombre}")
print(f"Área: {rectangulo.area()}")
print(f"Perímetro: {rectangulo.perimetro()}")
```

» Ejecución

```
Nombre: Circulo
Área: 78.54
Perímetro: 31.42
Nombre: Rectángulo
Área: 48
Perímetro: 28
```

3.- Crear las clases bases Nadador y Volador, heredar en las clases Pato y Cisne.

» Código

```python
class Nadador: # clase base 1
    def nadar(self):
        print("Nadando en el agua")

class Volador: # clase base 2
```

```
    def volar(self):
        print("Volando por el aire")

class Pato(Nadador, Volador):
    def graznar(self):
        print("¡Cuac!")

class Cisne(Nadador, Volador):
    def graznar(self):
        print("¡Graa Graa Graa!")

pato = Pato()
pato.nadar()
pato.volar()
pato.graznar()
cisne = Cisne()
cisne.nadar()
cisne.volar()
cisne.graznar()
```

» Ejecución

```
Nadando en el agua
Volando por el aire
¡Cuac!
Nadando en el agua
Volando por el aire
¡Graa Graa Graa!
```

4.- Crear la clase derivada IMC a partir de las clases Peso y Altura.

» Código

```
class Peso:
    def __init__(self, peso_kg):
        self.peso_kg = peso_kg

class Altura:
    def __init__(self, altura_m):
        self.altura_m = altura_m

class IMC(Peso, Altura):
    def __init__(self, peso_kg, altura_m):
        Peso.__init__(self, peso_kg)
        Altura.__init__(self, altura_m)

    def calcular_imc(self):
        if self.altura_m < 0:
            raise ValueError ("La altura debe ser mayor que 0")
        return self.peso_kg / (self.altura_m ** 2)

    def categoria_imc(self):
        imc = self.calcular_imc()
        if imc < 18.5:
            return "Bajo peso"
        elif imc < 25:
            return "Normal"
        elif imc < 30:
            return "Sobrepeso"
        else:
            return "Obesidad"

    def mostrar_resultado(self):
```

```
        imc = self.calcular_imc()
        categoria = self.categoria_imc()
        return f"IMC : {imc:.2f} - Categoría : {categoria}"

def leer_float(mensaje):
    while True:
        try:
            valor = float(input(mensaje))
            if valor <= 0:
                print("Por favor, ingrese un valor positivo")
                continue
            return valor
        except ValorError:
            print("Entrada invalida, ingrese un número valido")

peso = leer_float("Ingresa tu peso en kilogramos: ")
altura = leer_float("Ingresa tu peso en kilogramos: ")

persona = IMC(peso_kg = peso, altura_m = altura)
print(persona.mostrar_resultado())
```

» Ejecución

```
Ingresa tu peso en kilogramos: 64.5
Ingresa tu peso en kilogramos: 1.64
IMC : 23.98 - Categoría : Normal
```

5.- Calcular la hipotenusa de un triángulo rectángulo utilizando la herencia.

» Código

```
class Catetoa:
    def __init__(self, cateto_a):
        self.cateto_a = cateto_a

class Catetob:
    def __init__(self, cateto_b):
        self.cateto_b = cateto_b

class Hipotenusa(Catetoa, Catetob):
    def __init__(self, cateto_a, cateto_b):
        Catetoa.__init__(self, cateto_a)
        Catetob.__init__(self, cateto_b)

    def calcular_hipotenusa(self):
        if (self.cateto_a < 0) or (self.cateto_b < 0):
            raise ValueError("Los catetos deben ser mayors que 0")
        return (self.cateto_a ** 2 + self.cateto_b ** 2) ** 0.5

    def mostrar_resultado(self):
        hipotenusa = self.calcular_hipotenusa()
        return f"La hipotenusa es : {hipotenusa}"

def leer_float(mensaje):
    while True:
        try:
            valor = float(input(mensaje))
            if valor <= 0:
                print("Por favor, ingrese un valor positivo")
                continue
            return valor
        except ValorError:
            print("Entrada invalida, ingrese un número valido")
```

```
33
34 catetoa = leer_float("Ingrese el cateto a: ")
35 catetob = leer_float("Ingrese el cateto b: ")
36
37 triangulo = Hipotenusa(cateto_a = catetoa, cateto_b = catetob)
38 print(triangulo.mostrar_resultado())
```

» Ejecución

```
1 Ingrese el cateto a: 3
2 Ingrese el cateto b: 4
3 La hipotenusa es : 5.0
```

6.- Crear la clase PersonaMultirol que es herencia de las clases Persona, Trabajador y Estudiante.

» Código

```
1  class Persona:
2      def __init__(self, nombre, edad):
3          self.nombre = nombre
4          self.edad = edad
5
6      def presentarse(self):
7          print(f"Hola soy {self.nombre} y tengo {self.edad} años")
8
9  class Trabajador:
10     def __init__(self, profesion, salario):
11         self.profesion = profesion
12         self.salario = salario
13
14     def trabajar(self):
15         print(f"Estoy trabajando como {self.profesion} y gano ${self.salario}")
16
17 class Estudiante:
18     def __init__(self, carrera, universidad):
19         self.carrera = carrera
20         self.universidad = universidad
21
22     def estudiar(self):
23         print(f"Estudio {self.carrera} en la {self.universidad}")
24
25 class PersonaMultirol(Persona, Trabajador, Estudiante):
26     def __init__(self, nombre, edad, profesion, salario, carrera, universidad):
27         Persona.__init__(self, nombre, edad)
28         Trabajador.__init__(self, profesion, salario)
29         Estudiante.__init__(self, carrera, universidad)
30
31     def mostrar_informacion(self):
32         print("==== INFORMACIÓN DE LA PERSONA ====")
33         self.presentarse()
34         self.trabajar()
35         self.estudiar()
36
37 def main():
38     persona1 = PersonaMultirol(nombre = "Juanita", edad = 25,
39                                profesion = "Desarrollador de software",
40                                salario = 2500,
41                                carrera = "Ingeniería Estadística e Informática",
42                                universidad = "Universidad Nacional del Altiplano")
43     persona1.mostrar_informacion()
44
45 if __name__=="__main__":
46     main()
```

» Ejecución

```
==== INFORMACIÓN DE LA PERSONA ====
Hola soy Juanita y tengo 25 años
Estoy trabajando como Desarrollador de software y gano $2500
Estudio Ingeniería Estadística e Informática en la Universidad Nacional del
    ↪ Altiplano
```