Práctica Clases 1

```
class Personaje:
    pass

harry_potter = Personaje()
```

Práctica Clases 2

```
class Dinosaurio:
    pass

velociraptor = Dinosaurio()
tiranousaurio_rex = Dinosaurio()
braquiosaurio = Dinosaurio()
```

Práctica Clases 3

```
class PlataformaStreaming:
    pass

netflix = PlataformaStreaming()
hbo_max = PlataformaStreaming()
amazon_prime_video = PlataformaStreaming()
```

Práctica Atributos 1

```
class Casa:
    def __init__(self, color, cantidad_pisos):
        self.color = color
        self.cantidad_pisos = cantidad_pisos

casa_blanca = Casa("blanco", 4)
```

Práctica Atributos 2

```
class Cubo:
    caras = 6
    def __init__(self,color):
        self.color = color

cubo_rojo = Cubo("rojo")
```

Práctica Atributos 3

```
class Personaje:
    real = False

    def __init__(self, especie, magico, edad):
        self.especie = especie
        self.magico = magico
        self.edad = edad
```

```
8.
9. harry_potter = Personaje("humano", True, 17)
```

Práctica Métodos 1

```
1. class Perro:
2.     def ladrar(self):
3.         print("Guau!")
4.
5. pluto = Perro()
6. pluto.ladrar()
```

Práctica Métodos 2

```
1. class Mago:
2.     def lanzar_hechizo(self):
3.         print("¡Abracadabra!")
4.
5. merlin = Mago()
6. merlin.lanzar_hechizo()
```

Práctica Métodos 3

```
1. class Alarma:
2.     def postergar(self,cantidad_minutos):
3.         print(f"La alarma ha sido pospuesta {cantidad_minutos} minutos")
```

Práctica Tipos de Métodos 1

```
1. class Mascota:
2.     @staticmethod
3.     def respirar():
4.         print("Inhalar... Exhalar")
```

Práctica Tipos de Métodos 2

```
1. class Jugador:
2.     vivo = False
3.
4.     @classmethod
5.     def revivir(cls):
6.         cls.vivo = True
```

Práctica Tipos de Métodos 3

```
1. class Personaje:
2.     def __init__(self, cantidad_flechas):
3.         self.cantidad_flechas = cantidad_flechas
4.
5.     def lanzar_flecha(self):
6.         self.cantidad_flechas = self.cantidad_flechas-1
```

Práctica Herencia 1

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

class Alumno(Persona):
    pass
```

Práctica Herencia 2

```
class Mascota:
    def __init__(self, edad, nombre, cantidad_patas):
        self.edad = edad
        self.nombre = nombre
        self.cantidad_patas = cantidad_patas

class Perro(Mascota):
    pass

teo = Perro(6,"Teo",4)
```

Práctica Herencia 3

```
class Vehiculo:
    def acelerar(self):
        pass
    def frenar(self):
        pass

class Automovil(Vehiculo):
    pass
```

Práctica Herencia Extendida 1

```
class Padre():
    def trabajar(self):
        print("Trabajando en el Hospital")

    def reir(self):
        print("Ja ja ja!")

class Madre():
    def trabajar(self):
        print("Trabajando en la Fiscalía")

class Hija(Madre, Padre):
    pass
```

Práctica Herencia Extendida 2

```python
class Vertebrado():
    vertebrado = True

class Ave(Vertebrado):
    tiene_pico = True
    def poner_huevos(self):
        print("Poniendo huevos")

class Reptil(Vertebrado):
    venenoso = True

class Pez(Vertebrado):
    def nadar(self):
        print("Nadando")
    def poner_huevos(self):
        print("Poniendo huevos")

class Mamifero(Vertebrado):
    def caminar(self):
        print("Caminando")
    def amamantar(self):
        print("Amamantando crías")

class Ornitorrinco(Mamifero, Pez, Reptil, Ave):
    pass
```

Práctica Herencia Extendida 3

```python
class Padre():
    color_ojos = "marrón"
    tipo_pelo = "rulos"
    altura = "media"
    voz = "grave"
    deporte_preferido = "tenis"
    def reir(self):
        return "Jajaja"
    def hobby(self):
        return "Pinto madera en mi tiempo libre"
    def caminar(self):
        return "Caminando con pasos largos y rápidos"

class Hijo(Padre):
    def hobby(self):
        return "Juego videojuegos en mi tiempo libre"
```

Práctica Polimorfismo 1

```python
palabra = "polimorfismo"
lista = ["Clases", "POO", "Polimorfismo"]
tupla = (1, 2, 3, 80)

for dato in [palabra, lista, tupla]:
    print(len(dato))
```

Práctica Polimorfismo 2

```
class Mago():
    def atacar(self):
        print("Ataque mágico")

class Arquero():
    def atacar(self):
        print("Lanzamiento de flecha")

class Samurai():
    def atacar(self):
        print("Ataque con katana")

gandalf = Mago()
hawkeye = Arquero()
jack = Samurai()

personajes = [hawkeye, gandalf, jack]

for personaje in personajes:
    personaje.atacar()
```

Práctica Polimorfismo 3

```
class Mago():
    def defender(self):
        print("Escudo mágico")

class Arquero():
    def defender(self):
        print("Esconderse")

class Samurai():
    def defender(self):
        print("Bloqueo")

def personaje_defender(personaje):
    personaje.defender()
```

Práctica Métodos Especiales 1

```
class Libro():
    def __init__(self, titulo, autor, cantidad_paginas):
        self.titulo = titulo
        self.autor = autor
        self.cantidad_paginas = cantidad_paginas

    def __str__(self):
        return f'"{self.titulo}", de {self.autor}'
```

Práctica Métodos Especiales 2

```
class Libro():
    def __init__(self, titulo, autor, cantidad_paginas):
        self.titulo = titulo
        self.autor = autor
        self.cantidad_paginas = cantidad_paginas

    def __len__(self):
        return self.cantidad_paginas
```

Práctica Métodos Especiales 3

```
class Libro():
    def __init__(self, titulo, autor, cantidad_paginas):
        self.titulo = titulo
        self.autor = autor
        self.cantidad_paginas = cantidad_paginas

    def __del__(self):
        print(f'Libro eliminado')
```