

Introducción a la programación con Python

Polimorfismo

Alexis Rodríguez

Marcel Morán C

Esquema

- Orden de resolución de métodos (MRO)
- Polimorfismo en acción
- Sintaxis de polimorfismo
- Ejemplo de polimorfismo

Orden de resolución de métodos (MRO)

```
class Padre:  
    def __str__(self):  
        return 'Hola yo soy un objeto de clase  
Padre'
```

```
class Hijo(Padre):  
    def __str__(self):  
        return 'Hola yo soy un objeto de clase hijo'
```

```
class Nieto(Hijo):  
    def __str__(self):  
        return 'Hola yo soy un objeto de clase  
nieto'
```

Sobrescribir




Orden de resolución de métodos (MRO)

```
un_ejemplo_nieto = Nieto()
```

```
representation = un_ejemplo_nieto.__str__()  
print(representation)
```

```
Hola yo soy un objeto de clase nieto
```

Método dentro de
clase Nieto



Cómo accedemos a funciones que estén en clases superiores?

```
print(Nieto.__mro__)
```

```
(__main__.Nieto, __main__.Hijo, __main__.Padre, object)
```

Orden de resolución de métodos (MRO)


```
un_ejemplo_nieto = Nieto()

print(Nieto.__mro__)

(__main__.Nieto, __main__.Hijo, __main__.Padre, object)
```

```
representation_str = super(Nieto, un_ejemplo_nieto).__str__()
print(representation_str)
```


Método dentro de
clase Hijo



Hola yo soy un objeto de clase hijo

```
representation_str = super(Hijo, un_ejemplo_nieto).__str__()
print(representation_str)
```

Método dentro de
clase Padre



Hola yo soy un objeto de clase padre

Polimorfismo en acción

Miembro de un equipo



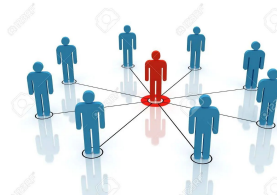
Hola yo soy
un miembro
de un equipo

Trabajador



Hola yo soy
un trabajador

Líder de un equipo



Sintaxis de polimorfismo en Python

```
class MiembroEquipo:
    def __init__(self, nombre equipo):
        self.nombre_equipo = nombre_equipo

    def describete(self):
        print(self.__str__())

    def __str__(self):
        return 'Soy un miembro de un equipo, el nombre de mi equipo es ' + self.nombre_equipo

class Trabajador:
    def __init__(self, salario, titulo_trabajo):
        self.salario = salario
        self.titulo_trabajo = titulo_trabajo

    def describete(self):
        print(self.__str__())

    def __str__(self):
        return 'Soy un trabajador, mi titulo de trabajo es ' + self.titulo_trabajo + ' y mi salario es ' + self.salario
```

Sintaxis de polimorfismo en Python

```
class LiderEquipo(MiembroEquipo, Trabajador):  
    def __init__(self, nombre equipo, salario, titulo_trabajo):  
        MiembroEquipo.__init__(self, nombre equipo)  
        Trabajador.__init__(self, salario, titulo_trabajo)  
  
    def __str__(self):  
        miembro equipo str =super(LiderEquipo, self).__str__()   
        trabajador equipo str =super(MiembroEquipo, self).__str__()   
        return miembro_equipo_str + '. Además, ' + trabajador_equipo_str
```

```
lider_ejemplo = LiderEquipo('Inteligencia Artificial', 2000, 'lider del equipo de Inteligencia Artificial')  
lider_ejemplo.describete()
```

Soy un miembro de un equipo, el nombre de mi equipo es Inteligencia Artificial.
Además, Soy un trabajador, mi título de trabajo es líder del equipo de Inteligencia Artificial y mi salario es 2000

Conclusión

- Polimorfismo se refiere a que hay varias maneras de ejecutar la misma acción
- Sobrecribir ocurre cuando un método en una clase hijo lleva el mismo nombre que una en la clase padre.
- No existe sobre carga de métodos en Python.
- El atributo `__mro__` permite acceder a la lista de búsqueda de atributos
- Usando la función `super()` podemos acceder a métodos dentro de superclases