funcionesMagicas

July 22, 2024

1 Ejemplos simples

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

    def __str__(self):
        return f'{self.nombre}, {self.edad} años'

    def __repr__(self):
        return f'Persona({self.nombre!r}, {self.edad!r})'

persona = Persona('Juan', 30)
print(persona)          # Usa __str__
print(repr(persona))    # Usa __repr__
```

```
Juan, 30 años
Persona('Juan', 30)
```

[ ]:
```python
class MiLista:
    def __init__(self, elementos):
        self.elementos = elementos

    def __len__(self):
        return len(self.elementos)

    def __getitem__(self, posicion):
        return self.elementos[posicion]

lista = MiLista([1, 2, 3, 4])
print(len(lista))    # Usa __len__
print(lista[2])      # Usa __getitem__
```

```
4
3
```

[ ]:
```python
class GestorRecursos:
    def __enter__(self):
        print("Entrando al contexto")
        return self

    def __exit__(self, tipo, valor, traza):
        print("Saliendo del contexto")

with GestorRecursos() as recurso:
    print("Dentro del bloque with")
```

```
Entrando al contexto
Dentro del bloque with
Saliendo del contexto
```

[ ]:
```python
class Saludador:
    def __init__(self, saludo):
        self.saludo = saludo

    def __call__(self, nombre):
        return f'{self.saludo}, {nombre}!'

saludador = Saludador('Hola')
print(saludador('Mundo'))  # Usa __call__
```

```
Hola, Mundo!
```

2 Ejemplos más avanzados

[ ]:
```python
class Singleton:
    _instancia = None

    def __new__(cls, *args, **kwargs):
        if cls._instancia is None:
            cls._instancia = super().__new__(cls)
        return cls._instancia

    def __init__(self, valor):
        self.valor = valor

obj1 = Singleton(10)
obj2 = Singleton(20)

print(obj1.valor)  # 20
print(obj2.valor)  # 20
print(obj1 is obj2)  # True
```

```
20
```

```
20
True
```

[ ]:
```python
class AtributosDinamicos:
    def __init__(self):
        self.datos = {}

    def __getattr__(self, nombre):
        return self.datos.get(nombre, f'{nombre} no encontrado')

    def __setattr__(self, nombre, valor):
        if nombre == 'datos':
            super().__setattr__(nombre, valor)
        else:
            self.datos[nombre] = valor

obj = AtributosDinamicos()
obj.nombre = 'Python'
print(obj.nombre)  # Python
print(obj.edad)    # edad no encontrado
```

```
Python
edad no encontrado
```

[ ]:
```python
class DiccionarioPersonalizado:
    def __init__(self):
        self.datos = {}

    def __setitem__(self, clave, valor):
        self.datos[clave] = valor

    def __getitem__(self, clave):
        return self.datos.get(clave, 'Clave no encontrada')

    def __delitem__(self, clave):
        if clave in self.datos:
            del self.datos[clave]

dic = DiccionarioPersonalizado()
dic['clave'] = 'valor'
print(dic['clave'])  # valor
del dic['clave']
print(dic['clave'])  # Clave no encontrada
```

```
valor
Clave no encontrada
```

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

    def __eq__(self, otro):
        return self.nombre == otro.nombre and self.edad == otro.edad

    def __lt__(self, otro):
        return self.edad < otro.edad

    def __hash__(self):
        return hash((self.nombre, self.edad))

p1 = Persona('Juan', 25)
p2 = Persona('Juan', 25)
p3 = Persona('Ana', 30)

print(p1 == p2)  # True
print(p1 < p3)   # True
print({p1, p2, p3})  # Conjunto con 2 elementos
```

```
True
True
{<__main__.Persona object at 0x7f2b18574b90>, <__main__.Persona object at
0x7f2b18575550>}
```

[ ]:
```python
class Optimizado:
    __slots__ = ['x', 'y']

    def __init__(self, x, y):
        self.x = x
        self.y = y

obj = Optimizado(1, 2)
print(obj.x, obj.y)
```

```
1 2
```

[ ]:
```python
class AccesoControlado:
    def __init__(self, valor):
        self.valor = valor

    def __getattribute__(self, nombre):
        print(f'Accediendo a {nombre}')
        return super().__getattribute__(nombre)

obj = AccesoControlado(10)
```

```
print(obj.valor)
```

```
Accediendo a valor
10
```

[ ]:
```
class Coleccion:
    def __init__(self, elementos):
        self.elementos = elementos

    def __contains__(self, item):
        return item in self.elementos

col = Coleccion([1, 2, 3])
print(2 in col)  # True
print(5 in col)  # False
```

```
True
False
```

[ ]:
```
class Contador:
    def __init__(self, inicio, fin):
        self.actual = inicio
        self.fin = fin

    def __iter__(self):
        return self

    def __next__(self):
        if self.actual < self.fin:
            actual = self.actual
            self.actual += 1
            return actual
        else:
            raise StopIteration

contador = Contador(0, 5)
for numero in contador:
    print(numero)
```

```
0
1
2
3
4
```

[ ]:
```
class Recurso:
    def __init__(self):
        print("Recurso creado")
```

```
    def __del__(self):
        print("Recurso liberado")

r = Recurso()
del r
```

```
Recurso creado
Recurso liberado
```