Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Docente:

Ing. Coyla Idme Leonel

Alumno:

Ticona Miramira Roberto Angel

Constructores y destructores

Constructores

Es un método especial que se ejecuta automaticamente cuando se crea ”instancia” un objeto de una clase.

Destructores

Es un método especial que se ejecuta automáticamente cuando el objeto está a punto de ser destruido.

Ciclo de vida de un objeto

El ciclo de vida de un objeto es el conjunto de etapas por las que pasa un objeto desde que se crea hasta que se destruye.

1. Creación: El objeto se construye e inicializa. (___init___)
2. Uso: Se llaman sus métodos y se accede a sus atributos.
3. Destrucción: El objeto se elimina de la memoria (___del___)

» Ejemplo 1

Crear el objeto TrianguloRectangulo y calcular la hipotenusa.

- Clase: TrianguloRectangulo.
- Atributo: Cateto_a, Cateto_b.
- Acción: calcular_hipotenusa(), destruir objeto.
- triangulo = TrianguloRectangulo(cateto1, cateto2)

Código

```
import math
class TrianguloRectangulo:
    def __init__(self, cateto_a, cateto_b):
        self.cateto_a = cateto_a
        self.cateto_b = cateto_b

    def calcular_hipotenusa(self):
        hipotenusa = math.sqrt(self.cateto_a ** 2 + self.cateto_b **2)
        return hipotenusa

    def __del__(self):
        print("Objeto TrianguloRectangulo destruido")

def main():
    try:
```

```
        cateto1 = float(input("Ingrese el valor del primer cateto: "))
        cateto2 = float(input("Ingrese el valor del segundo cateto: "))

        triangulo = TrianguloRectangulo(cateto1, cateto2)
        resultado = triangulo.calcular_hipotenusa()
        print(f"La hipotenusa del triángulo es {resultado:.2f}")
    except NameError:
        print("El objeto triangulo ya no existe (fue destruido)")
if __name__=="__main__":
    main()
```

Ejecución

```
Ingrese el valor del primer cateto: 3
Ingrese el valor del segundo cateto: 4
La hipotenusa del triángulo es 5.00
Objeto TrianguloRectangulo destruido
```

» Ejemplo 2

Crear el objeto Circulo y calcular el valor de su área.

- Clase: Circulo.
- Atributo: Radio.
- Acción: calcularArea()
- Objetos: circulo = Circulo()

Código

```
import math
class Circulo:
    def __init__(self, radio):
        self.radio = radio
        print("Objeto circulo creado")

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

radio_usuario = float(input("Ingrese el radio del circulo: "))
circulo = Circulo(radio_usuario)
rpta = circulo.calcularArea()
print(f"El área del circulo con radio {circulo.radio} es {rpta:.2f}")

del circulo

try:
    print(circulo)
except NameError:
    print("El objeto fue finiquitado")
```

Ejecución

```
Ingrese el radio del circulo: 3
Objeto circulo creado
El área del circulo con radio 3.0 es 28.27
El objeto fue finiquitado
```

» Ejemplo 3

Crear el objeto Comida y calcular las calorías totales.

- Clase: Comida.
- Atributos: Proteinas, Carbohidratos, Grasas.
- Acción: calcular_calorias(), mostrar_informacion()
- Objeto: almuerzo = Comida()

Código

```
class Comida:
    def __init__(self, proteinas, carbohidratos, grasas):
        self.proteinas = proteinas
        self.carbohidratos = carbohidratos
        self.grasas = grasas
        print("Comida creada")
        print(f"{self.proteinas} g. {self.carbohidratos} g. {self.grasas} g.")

    def calcular_calorias(self):
        calorias = self.proteinas * 4 + self.carbohidratos * 4 + self.grasas * 4
        return calorias

    def mostrar_informacion(self):
        print("INFORMACIÓN NUTRICIONAL")
        print(f"Proteínas: {self.proteinas} g.")
        print(f"Carbohidratos: {self.carbohidratos} g.")
        print(f"Grasas: {self.grasas} g.")
        print(f"Calorías totales: {self.calcular_calorias()} kcal.")

prot = float(input("Ingrese gramos de proteinas: "))
carb = float(input("Ingrese gramos de carbohidratos: "))
lip = float(input("Ingrese gramos de lípidos: "))

almuerzo = Comida(prot, carb, lip)
almuerzo.mostrar_informacion()

del almuerzo

try:
    almuerzo.mostrar_informacion()
except:
    print("Objeto almuerzo eliminado")
```

Ejecución

```
Ingrese gramos de proteinas: 30
Ingrese gramos de carbohidratos: 50
Ingrese gramos de lípidos: 20
Comida creada
30.0 g. 50.0 g. 20.0 g.
INFORMACIÓN NUTRICIONAL
Proteínas: 30.0 g.
Carbohidratos: 50.0 g.
Grasas: 20.0 g.
Calorías totales: 400.0 kcal.
Objeto almuerzo eliminado
```

» Ejemplo 4

Crear la clase estudiante con los atributos nombre, edad, carrera y utilizar un arreglo para almacenar varios estudiantes.

- Clase: Estudiante.
- Atributos: Nombre, Edad, Carrera.
- Acción: Mostrar_informacion.
- Objeto: estudiante.

Código

```
import gc
class Estudiante:
    def __init__(self, nombre, edad, carrera):
        self.nombre = nombre
        self.edad = edad
        self.carrera = carrera
        print(f"Estudiante registrado {self.nombre}. {self.edad} años
            ↪ {self.carrera}")

    def mostrar_informacion(self):
        print(f"{self.nombre} estudia {self.carrera} y tiene {self.edad} años")

    def __del__(self):
        print(f"Estudiante {self.nombre} eliminado")

datos_estudiantes = [("Ana", 20, "Medicina"),
                         ("Luis", 22, "Ingenieria"),
                         ("Carla", 19, "Arquitectura"),
                         ("Roberto", 23, "Nutrición")]

grupo = []

for datos in datos_estudiantes:
        estudiante = Estudiante(*datos)
        estudiante.mostrar_informacion()
        grupo.append(estudiante)

grupo.clear()
del estudiante
gc.collect
print("Fin del programa")
```

Ejecución

```
Estudiante registrado Ana. 20 años Medicina
Ana estudia Medicina y tiene 20 años
Estudiante registrado Luis. 22 años Ingenieria
Luis estudia Ingenieria y tiene 22 años
Estudiante registrado Carla. 19 años Arquitectura
Carla estudia Arquitectura y tiene 19 años
Estudiante registrado Roberto. 23 años Nutrición
Roberto estudia Nutrición y tiene 23 años
Estudiante Carla eliminado
Estudiante Luis eliminado
Estudiante Ana eliminado
Estudiante Roberto eliminado
Fin del programa
```

» Ejemplo 5

Crear la clase Libro con los atributos titulo, autor, anio y utilizar un arreglo para almacenar varios libros.

- Clase: Libro.

- Atributos: Titulo, Autor, Anio.
- Acción: Mostrar_informacion.
- Objeto: libro.

Código

```
import gc
class Libro:
    def __init__(self, titulo, autor, anio):
        self.titulo = titulo
        self.autor = autor
        self.anio = anio
        print(f"Libro registrado {self.titulo} de {self.autor} - {self.anio}")

    def mostrar_informacion(self):
        print(f"{self.titulo} fue escrito por {self.autor} en el año {self.anio}")

    def __del__(self):
        print(f"Libro {self.titulo} eliminado")

libros_datos = [("Cien años de soledad", "Gabriel García Marquez", 1967),
                ("1984", "George Orweli", 1949),
                ("Don Quijote de la Mancha", "Miguel de Cervantes", 1605)]

biblioteca = []

for datos in libros_datos:
    libro = Libro(*datos)
    libro.mostrar_informacion()
    biblioteca.append(libro)

biblioteca.clear()
del libro
gc.collect()
print("Fin de programa")
```

Ejecución

```
Libro registrado Cien años de soledad de Gabriel García Marquez - 1967
Cien años de soledad fue escrito por Gabriel García Marquez en el año 1967
Libro registrado 1984 de George Orweli - 1949
1984 fue escrito por George Orweli en el año 1949
Libro registrado Don Quijote de la Mancha de Miguel de Cervantes - 1605
Don Quijote de la Mancha fue escrito por Miguel de Cervantes en el año 1605
Libro 1984 eliminado
Libro Cien años de soledad eliminado
Libro Don Quijote de la Mancha eliminado
Fin de programa
```

» Ejemplo 6

Crear la clase Producto con los atributos nombre, precio, cantidad y almacenar en un arreglo.

- Clase: Producto.
- Atributos: Nombre, Precio, Cantidad.
- Acción: Mostrar_informacion.
- Objeto: producto.

```python
import gc
class Producto:
    def __init__(self, nombre, precio, cantidad):
        self.nombre = nombre
        self.precio = precio
        self.cantidad = cantidad
        print(f"\nProducto registrado {self.nombre} - $. {self.precio:.2f} en stock
            ↪ {self.cantidad}")

    def mostrar_informacion(self):
        print(f"{self.nombre} precio en $. {self.precio:.2f} en stock
            ↪ {self.cantidad}")

    def __del__(self):
        print(f"Producto eliminado: {self.nombre}")

producto_datos = [("Manzana", 0.5, 100),
                  ("Pan", 0.3, 50),
                  ("Leche", 3.5, 30)]

inventario = []

for datos in producto_datos:
    producto = Producto(*datos)
    producto.mostrar_informacion()
    inventario.append(producto)

inventario.clear()
del producto
gc.collect()
print("Fin de programa")
```

Ejecución

```
Producto registrado Manzana - $. 0.50 en stock 100
Manzana precio en $. 0.50 en stock 100

Producto registrado Pan - $. 0.30 en stock 50
Pan precio en $. 0.30 en stock 50

Producto registrado Leche - $. 3.50 en stock 30
Leche precio en $. 3.50 en stock 30
Producto eliminado: Pan
Producto eliminado: Manzana
Producto eliminado: Leche
Fin de programa
```

» Ejemplo 7

Crear la clase Curso con los atributos nombre, codigo, profesor y agregar datos a un arreglo.

- Clase: Curso.
- Atributos: Nombre, Codigo, Profesor.
- Acción: Mostrar_informacion.
- Objeto: curso.

Código

```python
import gc
class Curso:
```

```
    def __init__(self, nombre, codigo, profesor):
        self.nombre = nombre
        self.codigo = codigo
        self.profesor = profesor
        print(f"\nCurso {self.nombre} registrado | Código {self.codigo} | Docente
            ↪ {self.profesor}")

    def mostrar_informacion(self):
        print(f"Curso {self.nombre} con código {self.codigo} y docente
            ↪ {self.profesor}")

    def __del__(self):
        print(f"Curso {self.nombre} finiquitado")

alumnos_datos = [("Sistemas de gestión de base de datos I", "EST304", "Jose Panfilo
    ↪ Tito Lipa"),
                 ("Lenguajes de programación II", "EST305", "Leonel Coyla Idme"),
                 ("Programación numérica", "EST207", "Fred Torres Cruz"),
                 ]

registros = []

for datos in alumnos_datos:
    curso = Curso(*datos)
    curso.mostrar_informacion()
    registros.append(curso)

registros.clear()
del curso
gc.collect()
print("Fin de programa")
```

Ejecución

```
Curso Sistemas de gestión de base de datos I registrado | Código EST304 | Docente
    ↪ Jose Panfilo Tito Lipa
Curso Sistemas de gestión de base de datos I con código EST304 y docente Jose
    ↪ Panfilo Tito Lipa

Curso Lenguajes de programación II registrado | Código EST305 | Docente Leonel
    ↪ Coyla Idme
Curso Lenguajes de programación II con código EST305 y docente Leonel Coyla Idme

Curso Programación numérica registrado | Código EST207 | Docente Fred Torres Cruz
Curso Programación numérica con código EST207 y docente Fred Torres Cruz
Curso Lenguajes de programación II finiquitado
Curso Sistemas de gestión de base de datos I finiquitado
Curso Programación numérica finiquitado
Fin de programa
```