





Guía: Pruebas de la clase Persona en Python

Introducción

En esta guía realizaremos pruebas prácticas con la clase `Persona` que creamos previamente. Aprenderemos a crear objetos de esta clase y a simular operaciones como inserciones y eliminaciones, verificando cómo se comportan los atributos y los valores por defecto.

Paso 1: Editar el archivo de Persona

 Editamos el archivo Python llamado `persona.py` dentro de la carpeta de trabajo:

 **Ruta del archivo:** `capa_datos_persona/persona.py`

En este archivo escribiremos las pruebas de la clase `Persona`.

Paso 2: Validar la ejecución directa del script

Descripción

Agregamos la comprobación `if __name__ == '__main__':` para asegurarnos de que las pruebas solo se ejecuten cuando el archivo se ejecute directamente.

Código trabajado (capa_datos_persona/prueba_persona.py):

```
if __name__ == '__main__':  
    pass # Aquí irán las pruebas
```

Explicación:

- Esta validación asegura que las pruebas no se ejecuten si el archivo es importado desde otro módulo.

Paso 3: Crear el primer objeto Persona con todos los datos

Descripción

Creamos un objeto `persona1` proporcionando todos los valores: `id_persona`, `nombre`, `apellido`, y `email`.

Código trabajado (capa_datos_persona/prueba_persona.py):

```
from logger_base import log  
  
if __name__ == '__main__':  
    persona1 = Persona(1, 'Juan', 'Pérez', 'j.perez@mail.com')  
    log.debug(persona1)
```

Explicación:

- Importamos `log` desde `logger_base` para imprimir en nivel debug.
- Creamos `persona1` con valores completos.
- Usamos `log.debug()` para mostrar el contenido del objeto.

Paso 4: Simular inserción sin `id_persona`

Descripción

Creamos un objeto simulando una inserción (sin proporcionar `id_persona`), usando **argumentos por nombre**.

Código trabajado (capa_datos_persona/persona.py):

En primer lugar, modificamos el constructor de la clase Persona, para aceptar valores opcionales usando None:

```
def __init__(self, id_persona=None, nombre=None, apellido=None, email=None):
    self._id_persona = id_persona
    self._nombre = nombre
    self._apellido = apellido
    self._email = email
```

Con esto ya podemos crear objetos de tipo Persona sin proporcionar todos los valores del constructor. Ej:

```
# Simular un insert
persona1 = Persona(nombre='Juan', apellido='Perez', email='jperez@mail.com')
log.debug(persona1)
```

Explicación:

- No pasamos `id_persona`, por lo que usa el valor por defecto `None`.
- Proporcionamos solo `nombre`, `apellido` y `email`.
- Verificamos que `id_persona` sea `None` al imprimir.



Paso 5: Simular eliminación solo con `id_persona`



Descripción

Creamos un objeto simulando una eliminación, donde solo necesitamos el `id_persona`.

Código trabajado (capa_datos_persona/persona.py):

```
# Simular un delete
persona1 = Persona(id_persona=1)
log.debug(persona1)
```

Explicación:

- Solo pasamos `id_persona`, los demás atributos toman el valor por defecto `None`.
- Ideal para simular una eliminación donde solo necesitamos identificar el registro.



Sección final: Código completo de los archivos trabajados

 **Ruta y nombre del archivo de clase Persona:** capa_datos_persona/persona.py

```
from logger_base import log
```

```
class Persona:
```

```
    def __init__(self, id_persona=None, nombre=None, apellido=None, email=None):
```

```
        self._id_persona = id_persona
```

```
        self._nombre = nombre
```

```
        self._apellido = apellido
```

```
        self._email = email
```

```
    def __str__(self):
```

```
        return f'''
```

```
            Id Persona: {self._id_persona}, Nombre: {self._nombre},
```

```
            Apellido: {self._apellido}, Email: {self._email}
```

```
        '''
```

```
    @property
```

```
    def id_persona(self):
```

```
        return self._id_persona
```

```
    @id_persona.setter
```

```
    def id_persona(self, id_persona):
```

```
        self._id_persona = id_persona
```

```
    @property
```

```
    def nombre(self):
```

```
        return self._nombre
```

```
    @nombre.setter
```

```
    def nombre(self, nombre):
```

```
        self._nombre = nombre
```

```
    @property
```

```
    def apellido(self):
```

```
        return self._apellido
```

```
    @apellido.setter
```

```
    def apellido(self, apellido):
```

```
        self._apellido = apellido
```

```
    @property
```

```
    def email(self):
```

```
        return self._email

    @email.setter
    def email(self, email):
        self._email = email

if __name__ == '__main__':
    persona1 = Persona(1, 'Juan', 'Perez', 'jperez@mail.com')
    log.debug(persona1)
    # Simular un insert
    persona1 = Persona(nombre='Juan', apellido='Perez', email='jperez@mail.com')
    log.debug(persona1)
    # Simular un delete
    persona1 = Persona(id_persona=1)
    log.debug(persona1)
```



Conclusión

👉 En esta guía aprendimos a probar la clase `Persona` de manera práctica. Creamos objetos con diferentes combinaciones de parámetros para simular operaciones como inserciones y eliminaciones en una base de datos. También usamos el módulo de logging para mostrar la información de manera estructurada.

🚀 ¡Con estas pruebas confirmamos que nuestra clase `Persona` está lista para integrarse con otras partes de la aplicación!

🌟 Sigue adelante con tu aprendizaje 🚀, ¡el esfuerzo vale la pena!

¡Saludos! 👋

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)