





Guía: Creación de la clase Persona en Python

Introducción

En esta guía aprenderemos a crear una clase `Persona` en Python que nos permitirá modelar personas con atributos básicos. Definiremos los atributos principales, el método `__str__`, y los métodos `get` y `set` para cada propiedad. Esta clase servirá como base para la gestión de objetos de tipo persona dentro de una aplicación de manejo de datos.

Paso 1: Crear el archivo de la clase Persona

 Primero, creamos un nuevo archivo Python dentro del proyecto llamado `persona.py`, ubicado en la carpeta:

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

Aquí es donde escribiremos todo el código de la clase.

Paso 2: Definir la clase Persona

Descripción

Ahora definiremos la clase `Persona` con su método constructor `__init__`, donde inicializaremos los atributos `id_persona`, `nombre`, `apellido`, y `email`.

Código trabajado (capa_datos_persona/persona.py):

```
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
```

Explicación:

- Creamos la clase `Persona`.
 - El método `__init__` recibe los parámetros `id_persona`, `nombre`, `apellido` y `email`.
 - Los valores recibidos se asignan a los atributos privados (con prefijo `_`).
-

Paso 3: Agregar el método `__str__`

Descripción

Añadimos un método que permita mostrar los datos de una persona de manera legible cuando imprimamos un objeto.

Código trabajado (capa_datos_persona/persona.py):

```
def __str__(self):
    return f'''
        Id Persona: {self._id_persona}, Nombre: {self._nombre},
        Apellido: {self._apellido}, Email: {self._email}
    '''
```

Explicación:

- El método `__str__` devuelve una cadena que representa los datos del objeto.
 - Usamos f-strings para formatear el texto con los valores de los atributos.
-



Paso 4: Definir métodos `get` y `set` para los atributos



Descripción

Creamos los métodos de acceso y modificación para cada uno de los atributos privados.



Código trabajado (capa_datos_persona/persona.py):

```
@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
```



Explicación:

- Usamos el decorador `@property` para definir los métodos `get`.

- Con `@atributo.setter` definimos los métodos `set`.
 - Cada atributo privado tiene su propio `get` y `set` para controlar el acceso y la modificación.
-



Sección final: Código completo de los archivos trabajados



Aquí puedes agregar el código completo del archivo:



Ruta y nombre del archivo: `capa_datos_persona/persona.py`

```
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
```

✅ Conclusión

👉 En esta guía creamos paso a paso la clase `Persona`, definiendo su estructura, atributos, métodos `__init__`, `__str__`, y sus propiedades con `get` y `set`. Esta clase será muy útil como entidad base para manejar registros de personas en nuestra aplicación, asegurando encapsulamiento y fácil acceso a los datos.

🚀 ¡Listos para usar la clase `Persona` en futuras pruebas y módulos del proyecto!

✨ 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)