|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **--------------------------**  **HNET**  **--------------------------**  **EXAMEN ODOO**  **--------------------------** Edgar Josué Benedetto Godoy+504 3330-0171 | edgar.benedetto@unah.hnFrancisco Morazán, Tegucigalpa, Honduras | |  | | --- | | [Pregunta 1]  [Respuesta Pag.2-3] | | [Pregunta 2]  [Respuesta Pag.4-5] | | [Pregunta 3]  [Respuesta Pag.5-6] | | [Pregunta 4]  [Respuesta Pag.6-7] | | [Pregunta 5]  [Respuesta Pag.7-8] | | [Pregunta 6]  [Respuesta Pag.8-9] | | [Pregunta 7]  [Respuesta Pag.9-13] | |  | | Observaciones | | Editor de texto utilizado | | Visual Studio Code | | Servicio Cloud para  Jupyter-Notebooks | | Colab - Google | |  | |

1. ¿Qué componentes básicos debe de llevar un addon para que funcione en Odoo?

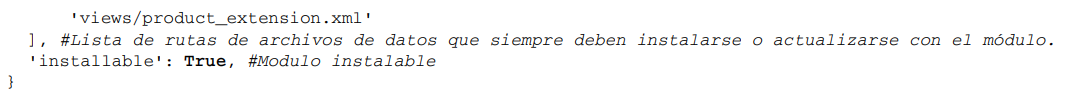
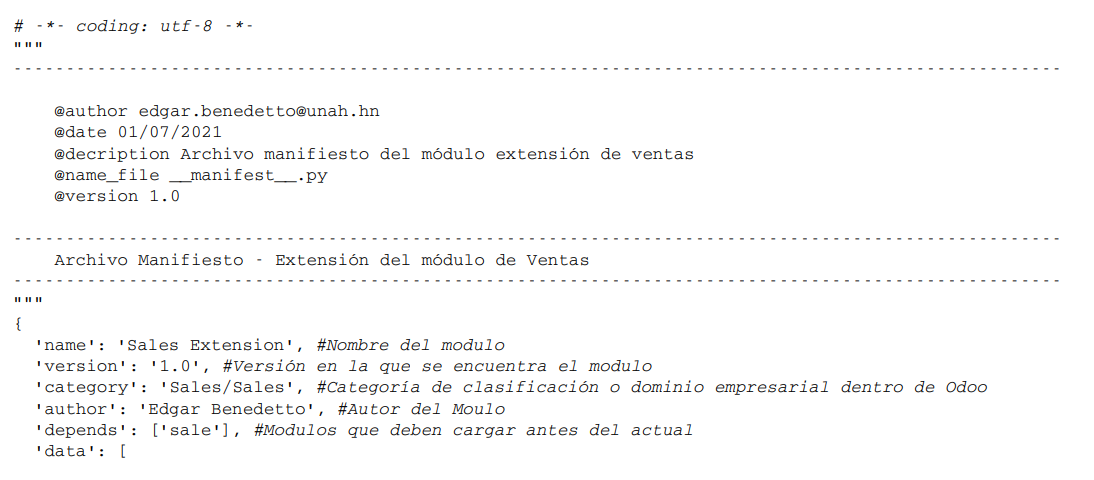
#### **R/** Componentes básicos de un addon para que funcione en Odoo

1. **Archivo Manifiesto**
   * Es un archivo en Python que sirve para especificar los metadatos del módulo contenidos en un diccionario y permite visualizar el módulo dentro de la lista de aplicaciones de Odoo. Se le da el nombre de **\_\_manifest\_\_.py.**

* Ejemplo:
  + Nombre del módulo: **sales\_extension**
  + Nombre del archivo:

**\_\_manifest\_\_.py**

* + Contenido del archivo:

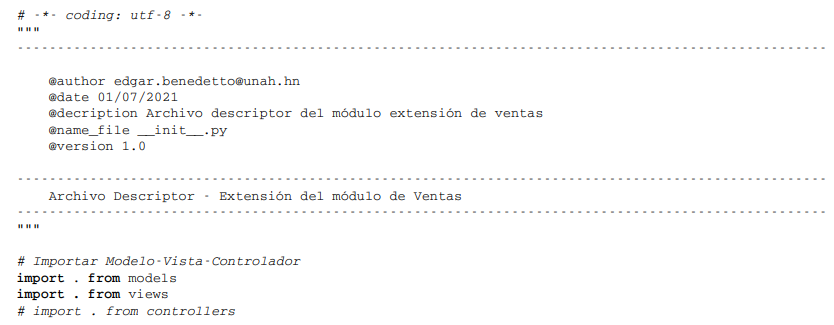


**2. Archivo Descriptor**

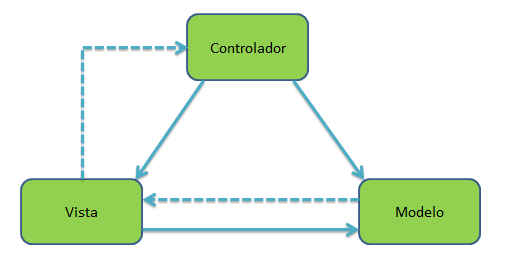
* + Sirve para cargar paquetes de Python en Odoo y también proporciona una descripción del módulo, tiene como característica que se ejecuta desde el inicio del programa. El archivo descriptor lleva el nombre de **\_\_init\_\_.py.**
* Ejemplo:
  + Nombre del módulo: **sales\_extension**
  + Nombre del archivo:

**\_\_init\_\_.py**

* + Contenido del archivo:



**3. Modelo-Vista-Controlador**

* +  En Odoo se hace uso del estilo de arquitectura de software que separa los datos de un módulo en tres componentes distintos.
  1. **El Modelo** que contiene la lógica de negocio y sus mecanismos de persistencia.
  2. **La Vista** o interfaz de usuario, es la capa de aplicación de los datos y se compone de la información que se envía al cliente junto a las formas de interactuar con dicha interfaz.
  3. **El Controlador** actúa de intermediario entre el Modelo y la Vista gestionando el flujo de información y transformación de los datos entre ellos.

2. **¿Como crear un modelo en Odoo?**

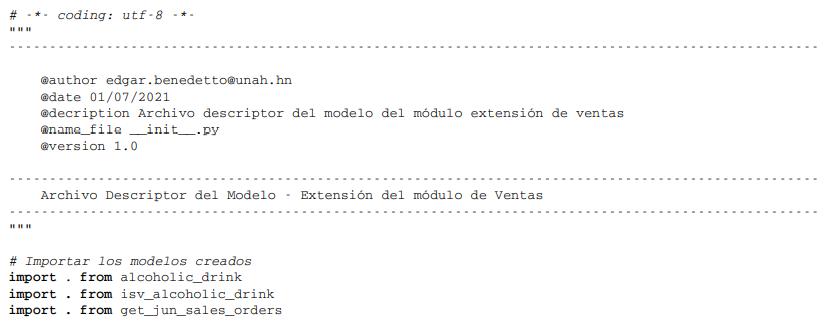
**R/** 1. Dentro de la carpeta del módulo al cual se le quiere crear un modelo, se debe crear una carpeta comunmente llamada: models

2. Dentro de la carpeta models se debe crear un archivo descriptor el cual lleva el nombre: **\_\_init\_\_.py.**

* Ejemplo:
  + Nombre del módulo: **sales\_extension**
  + Nombre del modelo: **alcoholic\_drink**
  + Nombre del archivo:

**\_\_init\_\_.py**

* + Contenido del archivo:

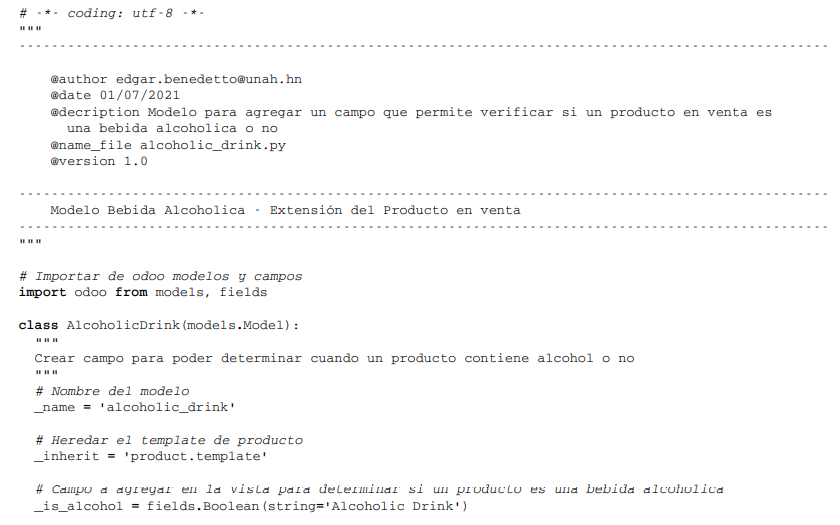


3. Crear los archivos en Python con nombres característicos para cada uno de los modelos necesarios.

* + En el modelo se debe escribir el atributo más importante el cual es **\_name** es **obligatorio** y define el **nombre del modelo** en el sistema **Odoo**.
  + Se puede **heredar** o usar cualquier módulo | objeto | clase | modelo | vista | un solo campo de módulos existentes. Algunos de los usos de la herencia son:
    1. **Cambiar** los atributos de algunos **campos** existentes.
    2. **Agregar** | **modificar** | **eliminar** **campos** antiguos o nuevos en el **modelo** existente | personalizado
    3. **Agregar** botones en la vista de **modelo** ya existente | personalizado
* Ejemplo:
  + Nombre del módulo: **sales\_extension**
  + Nombre del modelo: **alcoholic\_drink**
  + Nombre del archivo:

**alcoholic\_drink.py**

* + Contenido del archivo:



3. ¿Cuál es la sintaxis básica para heredar a una vista en Odoo?

R/

* La característica que permite interactuar con los objetos subyacentes sin modificar directamente el objeto original es la herencia, funciona como un agregado de capas para la modificación por encima de los objetos existentes.
* Las especificaciones de herencia se componen de un **localizador de elementos**, para que coincida con el elemento heredado en la vista principal, y el elemento secundario que se utilizará para modificar el elemento heredado. Hay tres tipos de localizadores de elementos para hacer coincidir un elemento de destino pero para heredar una vista se hace uso solamente de uno:
  1. Un elemento campo | field con un atributo nombre | name, la coincidencia se obtiene con el primero campo que tenga el mismo nombre, ignorando todos los demás atributos durante la comparación.
  2. Sintaxis básica para heredar una vista en Odoo

**<field name="inherit\_id" ref="library.view\_book\_form"/>**

* Ejemplo:
  + Nombre del módulo: **sales\_extension**
  + Nombre del modelo: **alcoholic\_drink**
  + Nombre del archivo:

**alcoholic\_drink.xml**

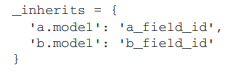
* + Contenido del archivo:



**4. ¿De qué manera se puede heredar un campo nuevo en un modelo existente dentro de Odoo?**

**R/**

* Para heredar solamente un campo en un modelo existente se hace uso del diccionario **\_inherits** el cual sirve para mapear el objeto padre.
* La llave del objeto **\_inherits** hace referencia al modelo y los valores hacen referencia a los campos del modelo posicionado en la llave.
* Implementa la herencia basada en la composición, el nuevo modelo expone todos los campos de los modelos heredados pero no almacena ninguno de ellos, los valores en sí permanecen almacenados en el registro vinculado.
* Estructura:



* Ejemplo:

****

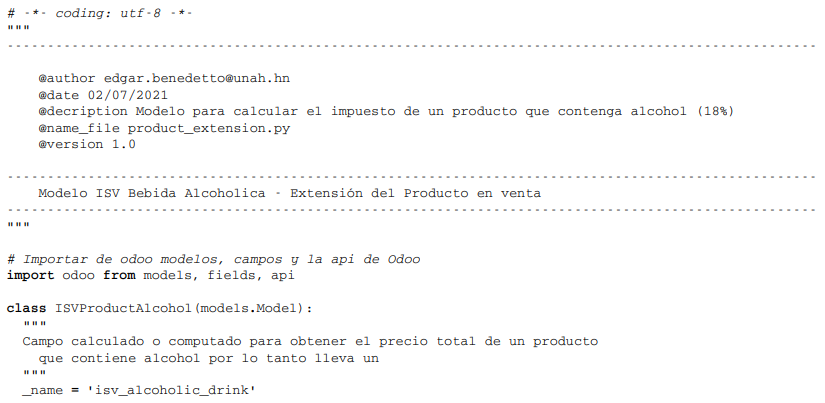
### **5. ¿Cuál es la sintaxis para crear un campo calculado?**

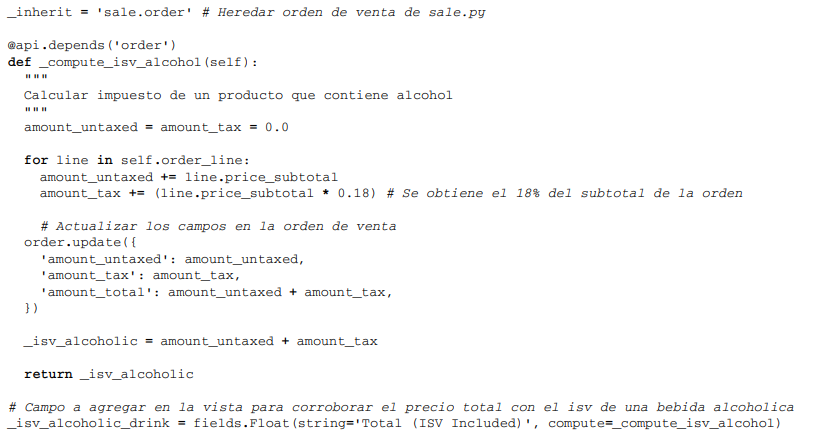
**R/**

* Los campos se pueden calcular (en lugar de leer directamente de la base de datos) utilizando el parámetro **compute**.
* Los campos calculados son de solo lectura de forma predeterminada. Si se usan los valores de otros campos, se deben especificar dichos campos usando **depends()**.
* Los campos calculados **no se almacenan de forma predeterminada**, se calculan y devuelven cuando se solicitan. La configuración store=True los almacenará en la base de datos y habilitará automáticamente la búsqueda.
* Ejemplo:
  + Nombre del módulo: **sales\_extension**
  + Nombre del modelo: **isv\_alcoholic\_drink**
  + Nombre del archivo:

**isv\_alcoholic\_drink.py**

* Contenido del archivo:



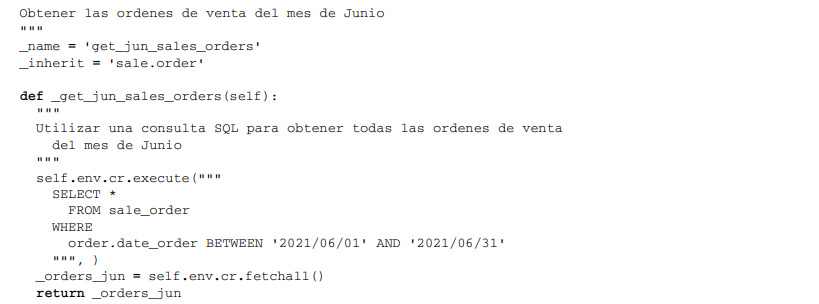
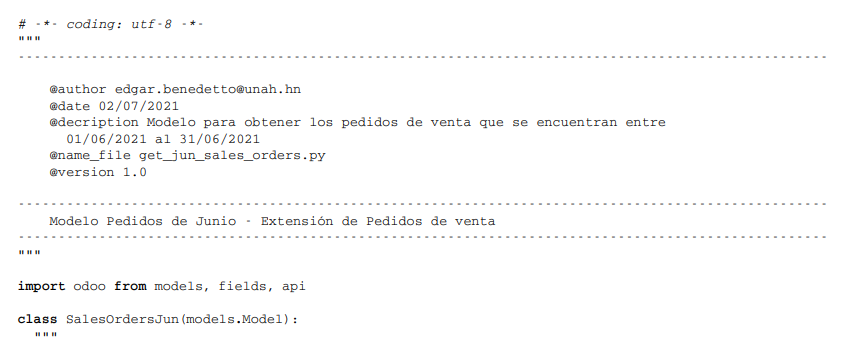
****

### **6. Buscar dentro de la tabla “sale.order” los pedidos de venta que se encuentren entre el 1/06/2021 al 31/06/2021. – Mostrar/escribir función correspondiente.**

* En algunas situaciones, donde el rendimiento es primordial, se tienen que ejecutar consultas SQL directamente.
* El entorno **(env)** contiene un atributo **cr**, es un **cursor** para la base de datos actual y permite ejecutar SQL directamente.
* Se puede imprimir la base de datos usando: **self.env.cr.dbname**, devuelve el **nombre de la base de datos actual**.
* Al no especificar que datos se requieren específicamente de la tabla **sale.order** se obtienen todos los datos de los pedidos de venta.
* Respuesta:
  + Nombre del módulo: **sales\_extension**
  + Nombre del modelo: **get\_jun\_sales\_orders**
  + Nombre del archivo:

**get\_jun\_sales\_orders.py**

* + Contenido del archivo:

****

### **7. Se le presenta el siguiente requerimiento: Se necesita saber cuántos productos por categoría asignada se encuentran dentro de un pedido de venta.**

1. **¿Qué proceso seguiría para obtener esa información?**
   * Un cliente de HNET solicita un apartado al final del documento de "Factura de cliente" en el cual se muestren los subtotales de cantidades pedidas para las siguientes categorías de producto: MAQUILLAJE, REPELENTES, ALCOHOL, ESMALTES Y OTROS. Por último, el cliente solicita un campo final con el total de todas las cantidades pedidas y contenidas dentro de la factura, el cual es la suma de todos los subtotales anteriores.
2. **¿Cómo agregarías esta información?**
3. **Notas:**
   * Categoría de producto = display\_name
   * Cantidades = product\_uom\_qty

### **Pasos a seguir para obtener y mostrar la información solicitada**

1. Crear o usar un módulo destinado a **extender** la **funcionalidad** del módulo encargado de la factura de una orden de venta. Implica crear la carpeta con un nombre característico: **"pos\_extension"**
2. Dentro de la carpeta del módulo **"pos\_extension" crear los archivos manifiesto y descriptor** del módulo incluyendo en el manifiesto en la sección de data todas las vistas utilizadas para mostrar la información solicitada
3. Crear el directorio **"models"** para contener todos los Modelos necesarios
4. Dentro de la carpeta **"models"** crear el **archivo descriptor** en el cual se importarán todos los modelos
5. Crear el **archivo** que contendrá el **modelo** para realizar los cambios solicitado por el cliente en la factura, designándole un nombre característico: **"category\_total.py"**
6. Crear el **directorio** encargado de contener las **vistas** necesarias para mostrar la información solicitada: **"views"**
7. Crear el archivo con extensión **".xml"** para **heredar** la **vista** de la **factura** y solamente agregarle los campos solicitados, designándole un nombre un nombre característico: **"category\_total.xml"**

### **Código**

1. Archivo Manifiesto del módulo **pos\_extension \_\_manifest\_\_.py.**
   * Nombre del módulo: **pos\_extension**
   * Nombre del archivo:

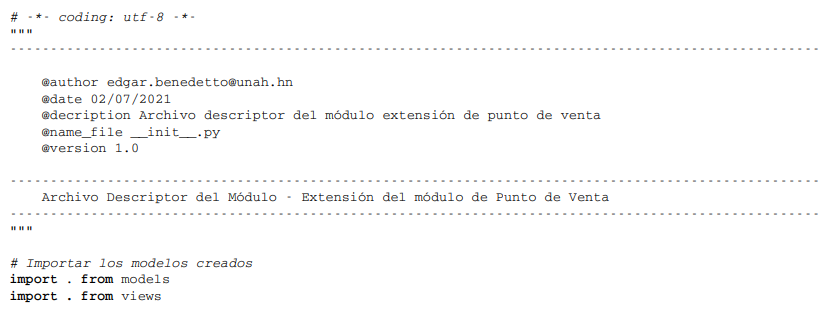
**\_\_manifest\_\_.py**

* +  Contenido del archivo:

1. Archivo Descriptor del módulo **pos\_extension \_\_init\_\_.py.**
   * Nombre del módulo: **pos\_extension**
   * Nombre del archivo:

**\_\_init\_\_.py**

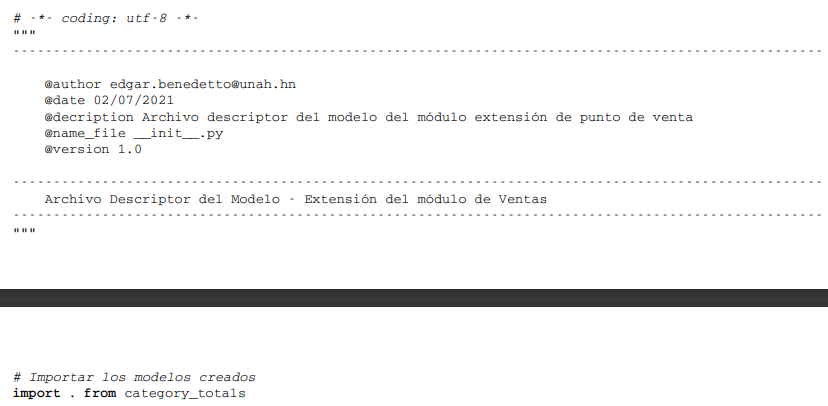
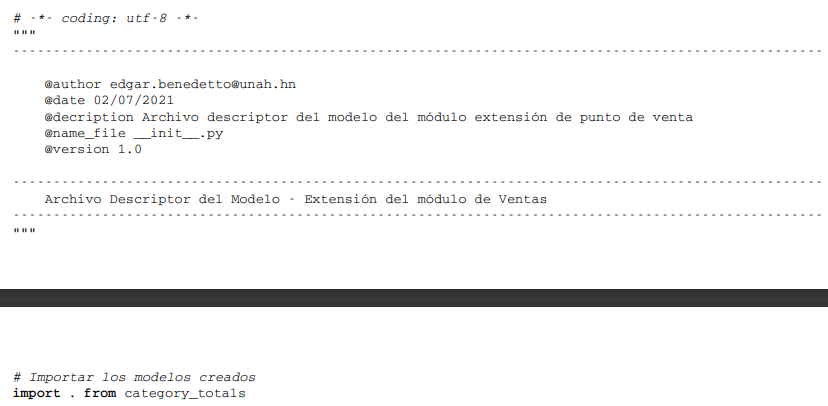
* + Contenido del archivo:

****

1. Dentro de la carpeta "models" crear el Archivo Descriptor del modelo correspondiente al módulo **pos\_extension \_\_init\_\_.py.**
   * Nombre del módulo: **pos\_extension**
   * Nombre del modelo: **category\_total**
   * Nombre del archivo:

**\_\_init\_\_.py**

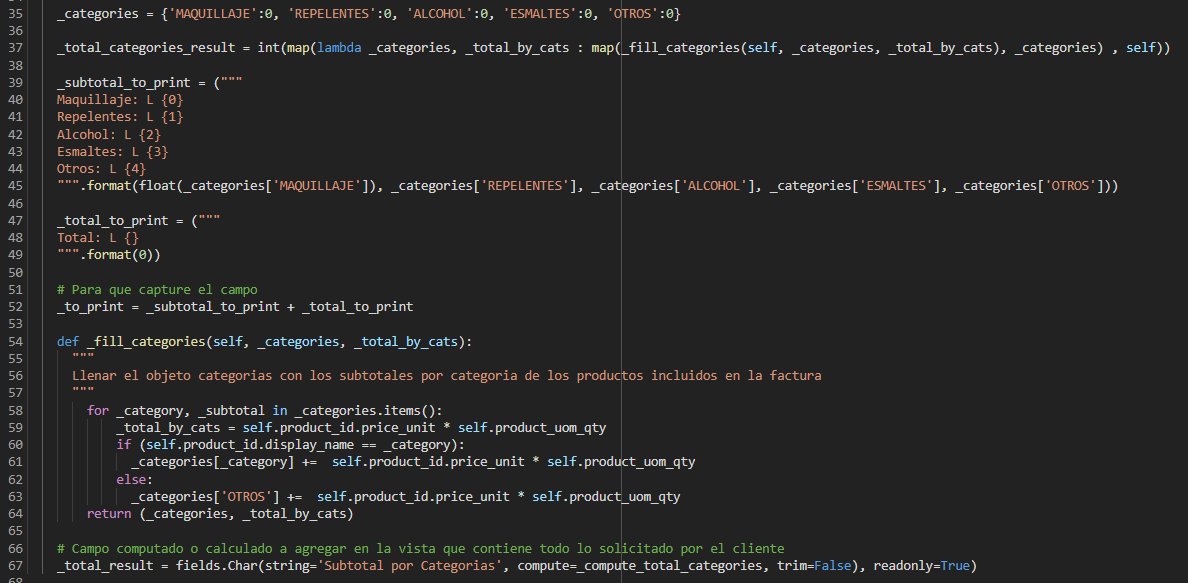
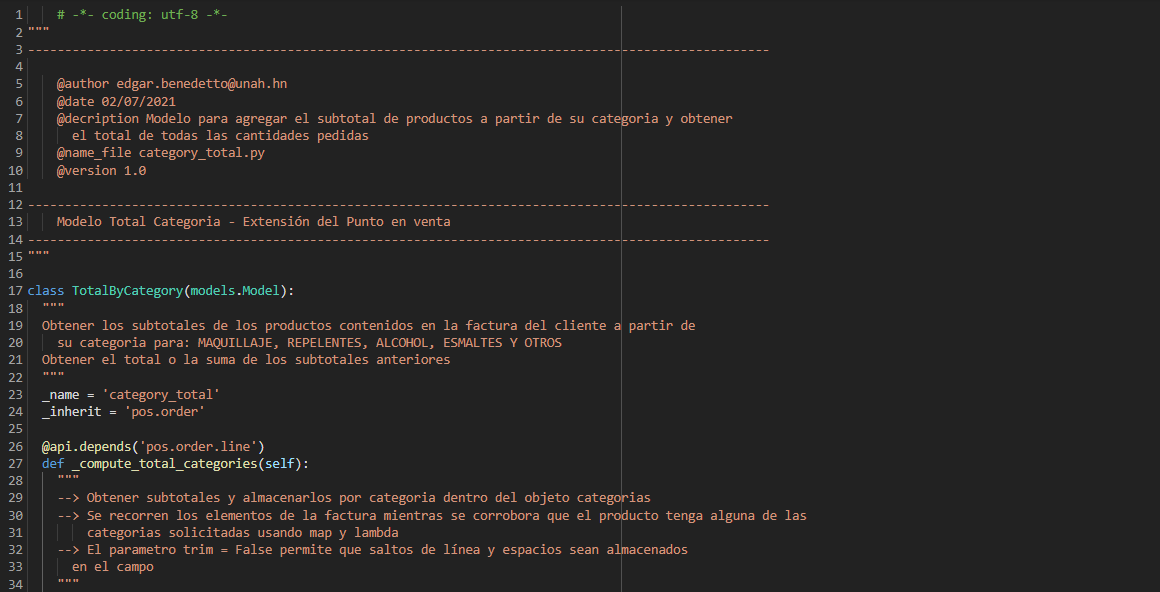
* + Contenido del archivo



1. Dentro de la carpeta "models" crear el Archivo del modelo **category\_total category\_total.py.**
   * Nombre del módulo: **pos\_extension**
   * Nombre del modelo: **category\_total**
   * Nombre del archivo:

**category\_total.py**

* + Contenido del archivo:

****

1. En el directorio "views" se debe crear el archivo: **category\_total.xml.**
   * Nombre del módulo: **sales\_extension**
   * Nombre del modelo: **category\_total**
   * Nombre del archivo:

**category\_total.xml**

* + Contenido del archivo:

