# Café Management System (with CSV storage)
File: `Cafe_Management_System.ipynb`
Description: A simple, menu-driven café management system implemented in Python for Jupyter Notebook.
This notebook includes comments and step-by-step explanations (suitable for submission).
It saves `menu.csv` and `orders.csv` automatically and loads saved data on startup.
## Prerequisites & How to run
1. Make sure you have Python 3 and Jupyter Notebook installed.
2. Put this notebook in a folder where you want `menu.csv` and `orders.csv` to be created.
3. Open the notebook in Jupyter and run each cell in order (Shift+Enter).
4. The main program starts automatically at the last cell — follow the text prompts in the output area.

Code:
```
# Imports and file handling functions
# These functions load and save the menu (menu.csv). If the file doesn't exist,
# a default menu is created and saved automatically.

import csv
import os

MENU_FILE = "menu.csv"
ORDERS_FILE = "orders.csv"

def load_menu():
"""Load menu from MENU_FILE. If file doesn't exist, create default menu and save
it."""
menu = {}
if os.path.exists(MENU_FILE):
with open(MENU_FILE, mode='r', newline='') as f:
reader = csv.reader(f)
header = next(reader, None) # skip header if present
for row in reader:
if row and len(row) >= 2:
item = row[0].strip()
try:
price = int(row[1])
except:
# skip rows with invalid price
continue
menu[item] = price
else:
# default menu (used only the first time)
menu = {
"Coffee": 50,
"Tea": 30,
"Sandwich": 80,
"Burger": 120,
"Pizza": 250,
"Pastry": 60
}
save_menu(menu)
return menu

def save_menu(menu):
"""Save the menu dictionary to MENU_FILE with a header."""
with open(MENU_FILE, mode='w', newline='') as f:
writer = csv.writer(f)
writer.writerow(["Item", "Price"])
for item, price in menu.items():
writer.writerow([item, price])

# Load menu into a global variable so other cells can access it
menu = load_menu()
print('Menu loaded. Current items:', list(menu.keys()))
```
Code:
```
# Functions to display the menu and take orders from the user (via input())
def show_menu(menu):
"""Print the menu in a readable format."""
print('\n------ Café Menu ------')
for item, price in menu.items():
print(f"{item:15} : ■{price}")
print('-----------------------')
```

```
def take_order(menu):
"""Interactively take an order using input(); returns a dictionary {item: qty}."
Prompts:
- Enter item name (case-insensitive)
- Enter quantity (integer)
- Type 'done' to finish ordering
"""
order = {}
show_menu(menu)
while True:
item = input("\nEnter item name to order (or type 'done' to finish): ").strip()
if item.lower() == 'done':
break
# Normalize to title case (to match menu keys)
item_title = item.title()
if item_title in menu:
while True:
try:
qty = int(input(f"Enter quantity for {item_title}: "))
if qty <= 0:
print("Quantity must be positive. Try again.")
continue
break
except ValueError:
print("Please enter a valid integer for quantity.")
order[item_title] = order.get(item_title, 0) + qty
else:
print("■ Item not found in menu. Please try again (check spelling).")

return order
```
Code:
```
# Billing and order-saving functions
def generate_bill(order, menu):
"""Prints the bill and returns the total amount."""
if not order:
print("No items in order.")
return 0
print('\n====== Café Bill ======')
total = 0
for item, qty in order.items():
price = menu[item] * qty
total += price
print(f"{item:15} x{qty:<3} = ■{price}")
print('-----------------------')
print(f"Total Amount = ■{total}")
print('=======================')
return total

def save_order(order, total, menu):
"""Append order details to ORDERS_FILE. Each item written as a row,
followed by a 'Bill Total' row and an empty separator row."""
file_exists = os.path.exists(ORDERS_FILE)
with open(ORDERS_FILE, mode='a', newline='') as f:
writer = csv.writer(f)
if not file_exists:
writer.writerow(["Item", "Quantity", "Price", "Total"])
for item, qty in order.items():
writer.writerow([item, qty, menu[item], qty * menu[item]])
writer.writerow(["---", "---", "Bill Total", total])
writer.writerow([]) # blank row as separator

# Example: generate_bill({'Coffee':2, 'Pizza':1}, menu)
```
Code:
```
# Function to add a new menu item and save it to CSV
def add_menu_item(menu):
"""Add a new item to the menu. Prompts for item name and price, saves menu to
file."""
item = input("Enter new item name: ").strip().title()
if not item:
print("Item name cannot be empty.")
return
if item in menu:
print("Item already exists in the menu.")
return
while True:
try:
```

```
price = int(input(f"Enter price for {item}: ■"))
if price <= 0:
print("Price must be positive. Try again.")
continue
break
except ValueError:
print("Please enter a valid integer for price.")
menu[item] = price
save_menu(menu)
print(f"{item} added successfully and saved to '{MENU_FILE}'!")
```

Code:

```
# Utility to preview saved orders (reads ORDERS_FILE if exists)
def show_saved_orders():
if not os.path.exists(ORDERS_FILE):
print('No orders recorded yet.')
return
import pandas as pd
try:
df = pd.read_csv(ORDERS_FILE)
display(df)
except Exception as e:
print('Could not read orders file:', e)
```

Code:

```
# Main program loop
def cafe_system():
global menu
print("\n===== Café Management System =====")
while True:
print("\n1. Show Menu")
print("2. Take Order")
print("3. Add Menu Item")
print("4. Show Saved Orders (preview)")
print("5. Exit")
choice = input("Enter your choice (1-5): ").strip()
if choice == '1':
show_menu(menu)
elif choice == '2':
order = take_order(menu)
if order:
total = generate_bill(order, menu)
save_order(order, total, menu)
print(f"Order saved to '{ORDERS_FILE}'.")
elif choice == '3':
add_menu_item(menu)
elif choice == '4':
show_saved_orders()
elif choice == '5':
print("Thank you for using the Café Management System. Goodbye! ■")
break
else:
print('Invalid choice. Enter a number between 1 and 5.')

# Run the system when the cell is executed
cafe_system()
```

## Final Notes
- `menu.csv` and `orders.csv` will be created in the same folder as this notebook.
- To reset the menu to default, delete `menu.csv` and re-run the imports/load cell.
- If you want a GUI later, request a `tkinter` or `streamlit` version and it can be provided.