Restaurant Food Order System

A Python-based command-line food ordering system that allows users to browse menus from different cuisines and place orders. This interactive application demonstrates menu management, order processing, and user input validation.

Features

- **Multi-Cuisine Support**: Italian and Indian food menus
- **Interactive Menu Display**: View available meals by cuisine type
- **Order Processing**: Select meals and specify quantities
- **Input Validation**: Verify meal availability before order confirmation
- **Order Summary**: Generate formatted order confirmations

Prerequisites

- Python 3.7 or higher
- No external dependencies required!

Setup

1. Clone the repository:

bash

```
git clone https://github.com/YOUR-USERNAME/food-order-system.git
cd food-order-system
```

2. No additional installation needed - uses Python standard library only!

Interactive Workflow

1. **Select Cuisine Type**

```
Welcome to the food order System!
Enter the type of food: Italian
```

2. **Browse Available Meals**

```
Available Italian Meals:
Pasta Bolognese
Pepperoni pizza
Margherita pizza
Lasagna
```

3. **Place Your Order**

Enter meal choice: Lasagna

Enter order quantity: 2

4. **Receive Confirmation**

You have ordered Lasagna 2

Project Structure

```
food-order-system/
│
├── Food Order System Project.ipynb    # Main Jupyter Notebook
├── food_order_system.py               # Standalone Python script
├── README.md                          # This file
├── requirements.txt                   # Python dependencies
└── LICENSE                            # License file
```

Functions

1. `find_meal(name, menu)`

Searches for a specific meal in a given menu.

Parameters:

- `name` (str): Name of the meal to find
- `menu` (list): List of available meals

Returns:

- `str`: The meal name if found
- `None`: If meal is not in the menu

Example:

python

```python
result = find_meal("Lasagna", italian_food)
# Returns: "Lasagna"


result = find_meal("Sushi", italian_food)
```

```
# Returns: None
```

Implementation:

python

```
def find_meal(name, menu):
    if name in menu:
        return name
    else:
        return None
```

---

2. `select_meal(name, food_type)`

Selects a meal from a specific cuisine menu.

Parameters:

- `name` (str): Name of the meal
- `food_type` (str): Type of cuisine ("Italian" or "Indian")

Returns:

- `str`: The meal name if found in the specified cuisine
- `None`: If meal is not found or invalid cuisine type

Example:

python

```
result = select_meal("Curry", "Indian")
# Returns: "Curry"


result = select_meal("Pizza", "Chinese")
# Returns: None
```

Logic Flow:

1. Check if food_type is "Italian" → search italian_food menu
2. Check if food_type is "Indian" → search indian_food menu
3. Otherwise → return None

---

3. `display_available_meals(food_type)`

Displays all available meals for a specific cuisine type.

Parameters:

- `food_type` (str): Type of cuisine ("Italian" or "Indian")

Returns:

- None (prints to console)

Output Examples:

Italian Menu:

```
Available Italian Meals:
Pasta Bolognese
Pepperoni pizza
Margherita pizza
Lasagna
```

Indian Menu:

```
Available Indian Meals:
Curry
Chutney
Samosa
Naan
```

Invalid Type:

```
Invalid food type
```

---

4. `create_summary(name, amount, food_type)`

Creates an order summary after validating the meal selection.

Parameters:

- `name` (str): Name of the meal ordered
- `amount` (str): Quantity ordered
- `food_type` (str): Type of cuisine

Returns:

- `str`: Order confirmation message or error message

Example:

python

```
summary = create_summary("Lasagna", "2", "Italian")
# Returns: "You have ordered Lasagna 2"


summary = create_summary("Sushi", "1", "Italian")
# Returns: "Meal not found"
```

Menu Items

Italian Cuisine

Item	Description
Pasta Bolognese	Classic Italian pasta with meat sauce
Pepperoni pizza	Pizza topped with pepperoni
Margherita pizza	Traditional pizza with tomato, mozzarella, and basil
Lasagna	Layered pasta with meat and cheese

Indian Cuisine

Item	Description
Curry	Spiced sauce dish
Chutney	Condiment/sauce
Samosa	Fried pastry with savory filling
Naan	Leavened flatbread

Key Programming Concepts

This project demonstrates:

1. **Lists**: Storing menu items in collections
2. **Functions**: Modular code organization
3. **Conditional Logic**: If-elif-else statements for decision making
4. **String Matching**: Checking if items exist in lists
5. **User Input**: Interactive console input/output
6. **String Formatting**: Using f-strings for output
7. **Return Values**: Functions returning different data types
8. **None Type**: Handling cases where data is not found

Learning Outcomes

Perfect for learning:

- Python functions and parameters
- List operations and membership testing
- Conditional statements (if-elif-else)
- User input handling
- String formatting
- Return values and None type
- Basic program flow control
- Debugging and testing

Educational Use

This project is ideal for:

- **Beginners**: Learning Python basics and function design
- **Students**: Understanding control flow and data structures
- **Self-learners**: Building practical applications
- **Educators**: Teaching modular programming

Contributing

Contributions are welcome! Here's how you can help:

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit your changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

Contribution Ideas:

- Fix the identified bugs
- Add more cuisines and dishes
- Implement a price system
- Add input validation
- Create unit tests
- Build a GUI version
- Add database support

License

This project is licensed under the MIT License - see the LICENSE file for details.

Acknowledgments

- Built as a learning project for Python programming
- Demonstrates basic restaurant ordering system concepts
- Perfect for beginners learning functions and conditionals

Contact

For questions, suggestions, or feedback, please open an issue on GitHub.