DESIGNING A SELF-ORDERING KIOSK PYTHON PROGRAM FOR ENHANCING CUSTOMER EXPERIENCE AT “TRAVELER'S TAVERN”

A CAPSTONE PROJECT DOCUMENTATION

College of Engineering and Architecture of

Mapúa Malayan Colleges Mindanao

Gen. Douglas MacArthur Highway, Davao City

In partial fulfillment of the requirements in

Computer Fundamentals and Programming 1 (Laboratory) Final Project

Submitted by:

Gene Marc Dacuycuy

Kherk Kenji Rheign Playda

Jahna Raya Royo

Section A141

Submitted to:

Prof. Emmy Grace Requillo

Computer Fundamentals and Programming 1 (Laboratory) Instructor

May 14, 2023

BUSINESS NAME

Traveler’s Tavern

BUSINESS TYPE

Restaurant Industry

BRIEF DESCRIPTION

Traveler's Tavern is a business that operates in the restaurant industry, offering a range of food and drink options with an international focus. This choice of industry reflects the group's interests in food and responds to the growing market demand for unique and authentic dining experiences. The name "Traveler's Tavern" was strategically chosen to evoke feelings of excitement and curiosity, appealing to customers who enjoy discovering new cultures and flavors. By combining a diverse menu with a welcoming atmosphere and adventurous branding, Traveler's Tavern seeks to create a memorable and enjoyable experience for all its patrons. With its featured self-ordering kiosk, the restaurant offers a convenient and efficient way for customers to place their orders, which reduces wait times and eliminates the need for traditional order-taking methods. This technology also allows customers to customize their orders according to their preferences, providing them with a greater sense of control over their dining experience. This business model highlights the importance of understanding customer needs and interests, as well as the value of creativity and innovation in the food service industry.

ALGORITHM

1. Start the program and welcome the user.
2. Ask the user to enter their name.
3. Prompt the user to choose between two options: dining in or takeout.
4. Display the menu items and their corresponding codes.
5. Ask the user to enter the item code of their desired order(s). If they want to order again, they would be prompted by a yes or no question.
6. Calculate the total cost of the order(s).
7. Ask the user to select their payment method: cash or e-money.
8. If the user chooses e-money, display a QR code for them to scan and proceed to the cashier register for confirmation. If they choose to pay with cash, prompt them to proceed to the cashier register.
9. Generate a receipt that includes the order number, total cost, VAT, and whether it's for dine-in or takeout.
10. Display a message thanking the user for their patronage and informing them that their order is being prepared.

PSEUDOCODE

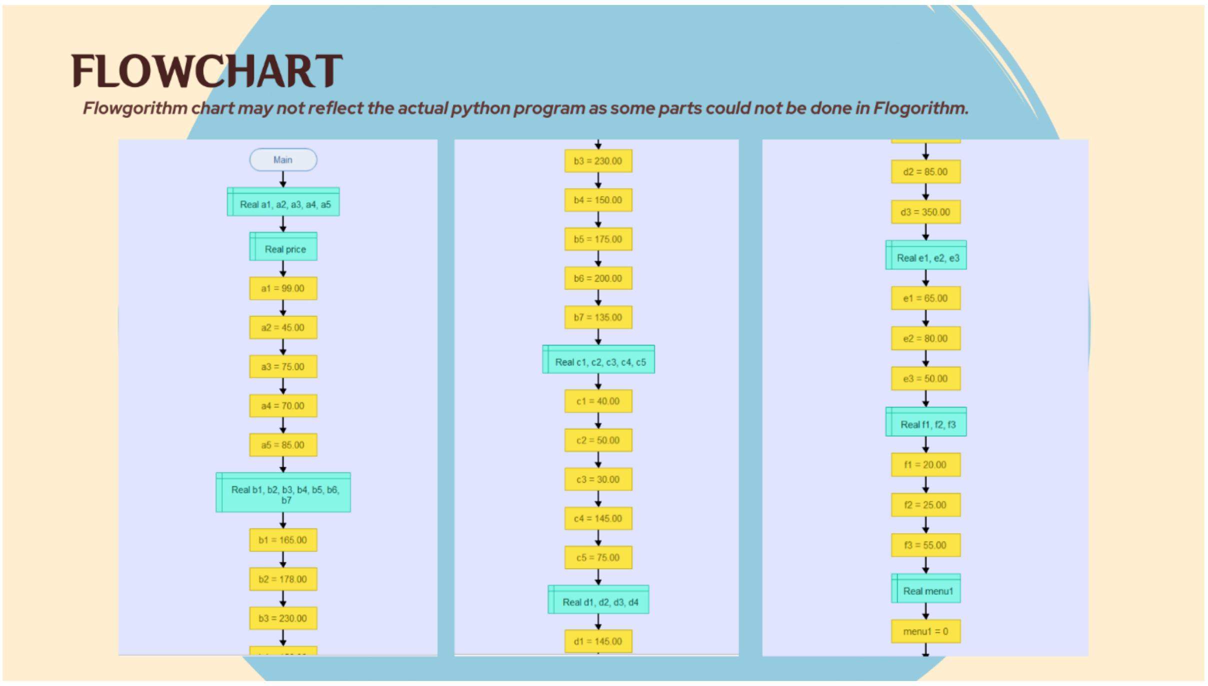
```
Display "Ad Astra Abyssosque!✨ Welcome to Traveler's Tavern!"
Ask the customer to enter their name
Read customer name as customerName

Display "Hello, " + customerName + "! Are you dining in or taking out?"
Initialize num_invalid_options as 0

While True:
    Display "Enter 'D' for dine-in or 'T' for takeout:"
    Read option

    If option.upper() not in ['D', 'T']:
        Increment num_invalid_options by 1
        Display "Invalid choice."

        If num_invalid_options == 3:
            Display "Too many invalid choices. Please wait 10 seconds
before trying again."
            Wait for 10 seconds
            Reset num_invalid_options to 0

        Continue to next iteration
    Else:
        Break the loop

If option.upper() == 'D' or 'Dine-In':
    Display "You have selected dine-in."
Else If option.upper() == 'T' or 'Takeout':
    Display "You have selected takeout."

For each category, items in the menu:
    Display category

    For each item_code, item_info in items:
        Display item_code, item_info['name'], item_info['price']
```

```
Initialize order_list as empty list

While True:
    Display "Please enter the item code of your order(s):"
    Read order
    Extend order_list with order split by "," and " "

    Display "Do you want to add more orders? (y/n)"
    Read order_more

    If order_more.lower() != "y":
        Break the loop

Initialize total_cost as 0

For each item_code in order_list:
    Convert item_code to uppercase

    For each category, items in the menu:
        Get the item_details with the item_code from items

        If item_details exists:
            Display item_details['name'], item_details['price']
            Add item_details['price'] to total_cost
            Break the loop
        Else:
            Continue to next iteration

Display "Total cost: PHP" + total_cost

Display "How do you want to pay? (cash/e-money)"
Read payment_method

If payment_method.lower() == "e-money":
    Display "Please scan the QR code and proceed to the cashier."
    Generate QR code
    Display the QR code
    Display "Type 'done' once you have scanned the QR code:"
    Read done

    If done.lower() == 'done':
        Display "Payment received. Thank you!"
        Display "Enter amount of cash: PHP"
        Read cash_received
        Calculate change as cash_received - total_cost
    Else:
        Display "Payment not received. Please ask assistance to our staff."
Else:
    Display "Please proceed to the cashier."

    While True:
        Display "Enter amount of cash: PHP"
        Read cash_received

        If cash_received is not a valid number:
            Display "Invalid input. Please enter a valid number."
            Continue to next iteration
        Else:
            Break the loop
    Calculate change as cash_received - total_cost

    If change < 0:
        Display "Insufficient payment. Please pay the correct amount."
    Else:
        Display "Change: PHP" + change

Print the receipt using the function print_receipt()
Display "Please wait as we prepare your meal! ✨"
```

FLOWCHART

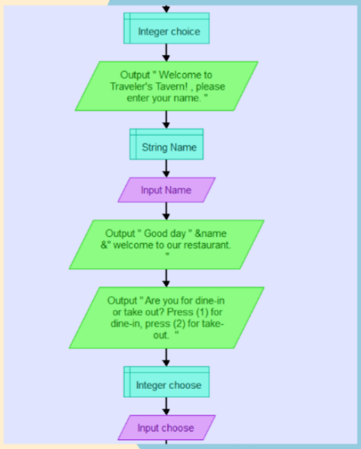
Flowgorithm chart may not reflect the actual python program as some parts could not be done in Flogorithm.

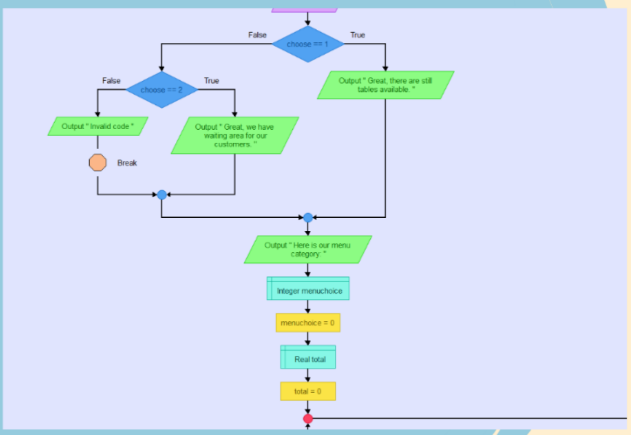
(Order is from right to left)



FLOWCHART

Flowgorithm chart may not reflect the actual python program as some parts could not be done in Flogorithm.

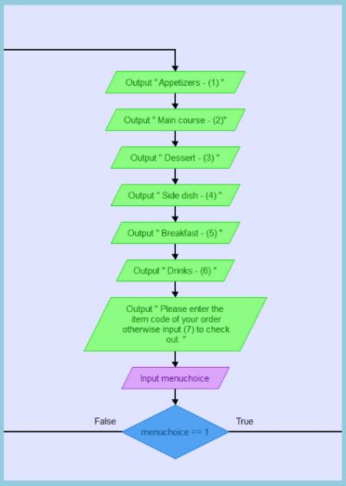

Integer choice
Output " Welcome to Traveler's Tavern! , please enter your name. "
String Name
Input Name
Output " Good day " &name &" welcome to our restaurant. "
Output " Are you for dine-in or take out? Press (1) for dine-in, press (2) for take-out. "
Integer choose
Input choose

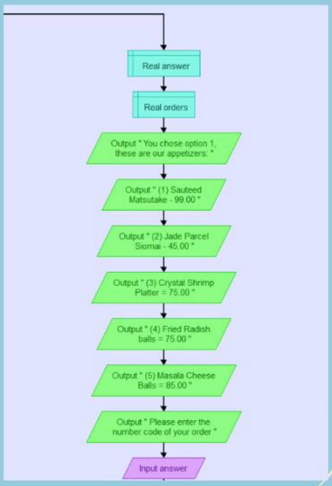

False
choose == 1
True
False
choose == 2
True
Output " Great, there are still tables available. "
Output " Invalid code "
Output " Great, we have waiting area for our customers. "
Break
Output " Here is our menu category: "
Integer menuchoice
menuchoice = 0
Real total
total = 0

FLOWCHART

Flowgorithm chart may not reflect the actual python program as some parts could not be done in Flogorithm.

TRUE path

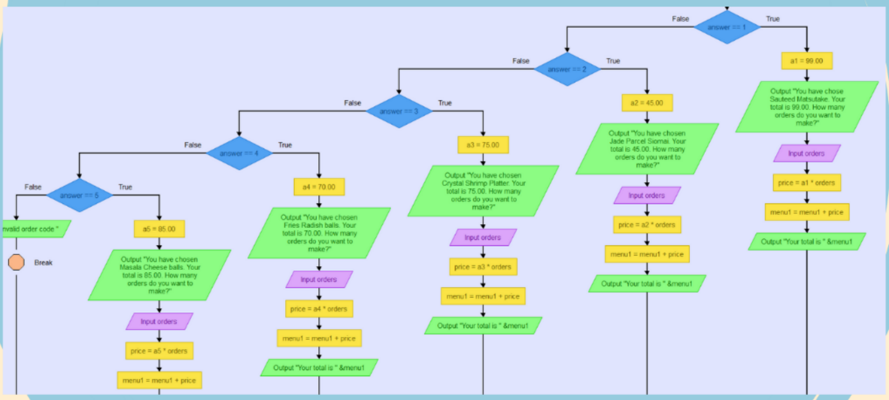

Output " Appetizers - (1) "
Output " Main course - (2)"
Output " Dessert - (3) "
Output " Side dish - (4) "
Output " Breakfast - (5) "
Output " Drinks - (6) "
Output " Please enter the item code of your order otherwise input (7) to check out. "
Input menuchoice
False
menuchoice == 1
True


Real answer
Real orders
Output " You chose option 1, these are our appetizers: "
Output " (1) Sauteed Matsutake - 99.00 "
Output " (2) Jade Parcel Siomai - 45.00 "
Output " (3) Crystal Shrimp Platter = 75.00 "
Output " (4) Fried Radish balls = 75.00 "
Output " (5) Masala Cheese Balls = 85.00 "
Output " Please enter the number code of your order "
Input answer

FLOWCHART

Flowgorithm chart may not reflect the actual python program as some parts could not be done in Flogorithm.

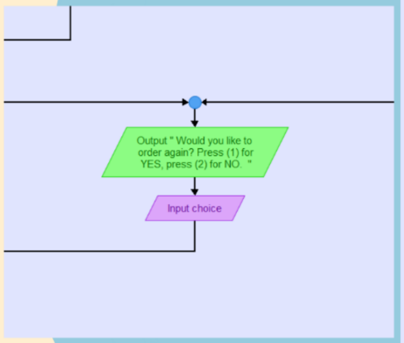
TRUE path

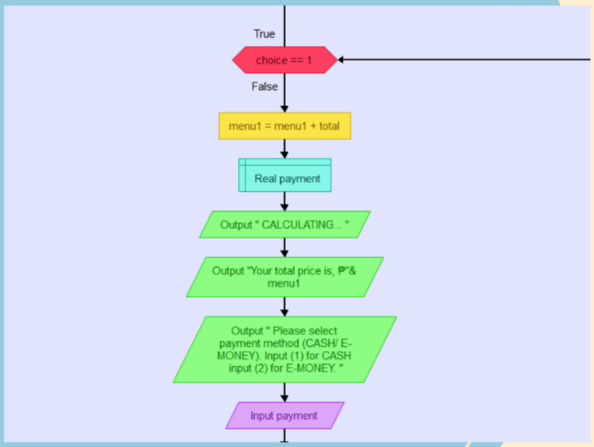

False
answer == 1
True
a1 = 99.00
Output "You have chose Sauteed Matsutake. Your total is 99.00. How many orders do you want to make?"
Input orders
price = a1 * orders
menu1 = menu1 + price
Output "Your total is " &menu1
False
answer == 2
True
a2 = 45.00
Output "You have chosen Jade Parcel Siomai. Your total is 45.00. How many orders do you want to make?"
Input orders
price = a2 * orders
menu1 = menu1 + price
Output "Your total is " &menu1
False
answer == 3
True
a3 = 75.00
Output "You have chosen Crystal Shrimp Platter. Your total is 75.00. How many orders do you want to make?"
Input orders
price = a3 * orders
menu1 = menu1 + price
Output "Your total is " &menu1
False
answer == 4
True
a4 = 70.00
Output "You have chosen Fries Radish balls. Your total is 70.00. How many orders do you want to make?"
Input orders
price = a4 * orders
menu1 = menu1 + price
Output "Your total is " &menu1
False
answer == 5
True
a5 = 85.00
Output "You have chosen Masala Cheese balls. Your total is 85.00. How many orders do you want to make?"
Input orders
price = a5 * orders
menu1 = menu1 + price
nvalid order code "
Break

FLOWCHART

Flowgorithm chart may not reflect the actual python program as some parts could not be done in Flogorithm.

TRUE path, if user chooses NO

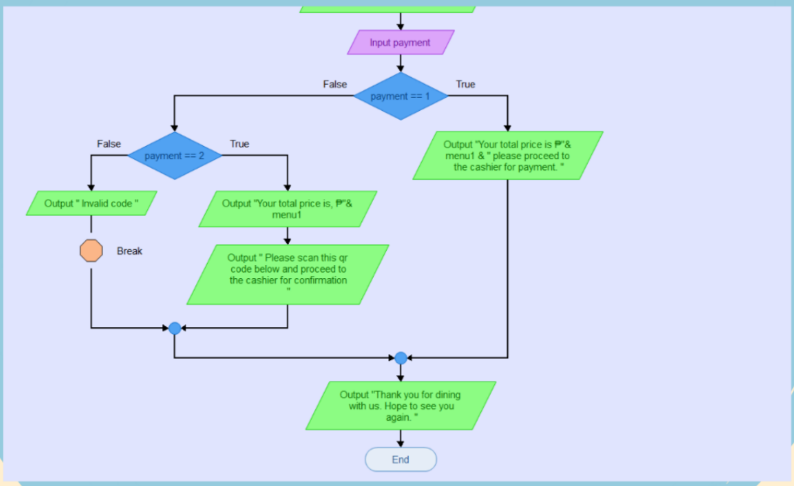

Output " Would you like to order again? Press (1) for YES, press (2) for NO. "
Input choice


True
choice == 1
False
menu1 = menu1 + total
Real payment
Output " CALCULATING... "
Output "Your total price is, ₱"& menu1
Output " Please select payment method (CASH/ E-MONEY). Input (1) for CASH input (2) for E-MONEY. "
Input payment

FLOWCHART

Flowgorithm chart may not reflect the actual python program as some parts could not be done in Flogorithm.

TRUE path, if user chooses NO


Input payment
False
payment == 1
True
Output "Your total price is ₱"& menu1 & " please proceed to the cashier for payment. "
False
payment == 2
True
Output " Invalid code "
Output "Your total price is, ₱"& menu1
Break
Output " Please scan this qr code below and proceed to the cashier for confirmation "
Output "Thank you for dining with us. Hope to see you again. "
End

COMPLETE SOURCE CODE

CPE_Final.py

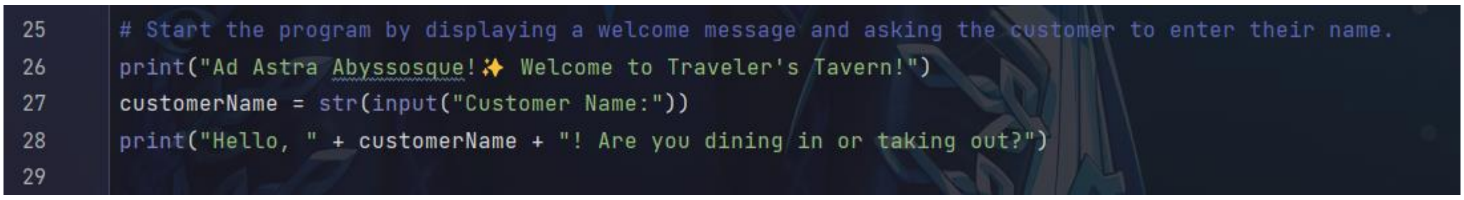
```
from menu_dict import menu
import time
import random
from printreceipt import print_receipt

business_name = "Traveler's Tavern"
order_number = random.randint(1000, 9999)
cashier = random.choice(['Diluc', 'Charles', 'Sara', 'Xiangling', 'Smiley
Yanxiao'])

# Start the program by displaying a welcome message and asking the customer to
enter their name.
print("Welcome to Traveler's Tavern!")
customerName = str(input("Customer Name:"))
print("Hello, " + customerName + "! Are you dining in or taking out?")

# Then the computer asks if they are dining in or taking out.
# It will repeat the question 3 times if the customer inputs an invalid option.
# If the customer enters an invalid dining option after 3 attempts, there will be a
10-second timer before using again.

num_invalid_options = 0

while True:
    option = input("Enter 'D' for dine-in or 'T' for takeout:  ")
    if option.upper() not in ['D', 'T']:
        num_invalid_options += 1
        print("Invalid choice.")
        if num_invalid_options == 3:
            print("Too many invalid choices. Please wait 10 seconds before trying
again.")
            time.sleep(10)
            num_invalid_options = 0
        continue
    else:
        break

if option.upper() == 'D' or 'Dine-In':
    print("You have selected dine-in.")
elif option.upper() == 'T' or 'Takeout':
    print("You have selected takeout.")

for category, items in menu.items():
    print(f"\n{category}:")
    for item_code, item_info in items.items():
        name = item_info['name']
        price = item_info['price']
        print(f"{item_code}: {name} - PHP {price}")

order_list = []
while True:
    order = input("Please enter the item code of your order(s): ")
    order_list.extend(order.split("," and ", "))
    order_more = input("Do you want to add more orders? (y/n) ")
    if order_more.lower() != "y":
        break

total_cost = 0
for item_code in order_list:
    item_code = item_code.upper()
    for category, items in menu.items():
        item_details = items.get(item_code)
        if item_details:
            print('')
            print(f"{item_details['name']} - PHP {item_details['price']:.2f}")
            total_cost += item_details['price']
            break
    else:
        print(f"Invalid item code: {item_code}")
print('')
print(f"Total cost: PHP {total_cost:.2f}")

payment_method = input("How do you want to pay? (cash/e-money) ")
if payment_method.lower() == "e-money":
    print("Please scan the QR code and proceed to the cashier.")
    # simulate QR code generation and scanning
    qr_code = """
        ######
        ##  ##
        ##  ##
        ##  ##
        ######
    """
    print(qr_code)
    done = input("Type 'done' once you have scanned the QR code: ")
    if done.lower() == 'done':
        print("Payment received. Thank you!")
        cash_received = float (input("Enter amount of cash: PHP "))
        change = cash_received - total_cost
    else:
        print("Payment not received. Please ask assistance to our staff.")

# If user uses the cash choice; a cashier transaction simulation is made.

else:
    print ("Please proceed to the cashier.")
    while True:
        try:
            cash_received = float(input("Enter amount of cash: PHP "))
            break
        except ValueError:
            print("Invalid input. Please enter a valid number.")

change = cash_received - total_cost

if change < 0:
    print ("Insufficient payment. Please pay the correct amount.")
else:
    print (f"Change: PHP {change:.2f}. Thank you!")

# printing the receipt
print_receipt (order_number, customerName, cashier, option, payment_method,
order_list, total_cost, cash_received,
              change)
```

menu_dict.py

```
menu = {
    'Appetizers': {
        'A1': {'name': 'Sauteed Matsutake', 'price': 99.00},
        'A2': {'name': 'Jade Parcel Siomai', 'price': 45.00},
        'A3': {'name': 'Crystal Shrimp Platter', 'price': 75.00},
        'A4': {'name': '4pc-Fried Radish Balls', 'price': 70.00},
        'A5': {'name': 'Masala Cheese Balls', 'price': 85.00}},

    'Main Course': {
        'M1': {'name': 'Zhongyuan Chop Suey', 'price': 165.00},
        'M2': {'name': 'Sticky Honey Roast', 'price': 178.00},
        'M3': {'name': 'Sweet Madame', 'price': 230.00},
        'M4': {'name': 'Avidya Goulash', 'price': 150.00},
        'M5': {'name': 'Qince Stir-fried Rice', 'price': 175.00},
        'M6': {'name': "Watcher's Biryani", 'price': 200.00},
        'M7': {'name': 'Setekh Potato Curry', 'price': 135.00}},

    'Dessert': {
        'D1': {'name': 'Almond Taho', 'price': 40.00},
        'D2': {'name': 'Padisarah Ube Pudding', 'price': 50.00},
        'D3': {'name': 'Tri-color Dango Mochi', 'price': 30.00},
        'D4': {'name': 'Halo-Halo', 'price': 145.00},
        'D5': {'name': 'Taiyaki Ice Cream', 'price': 75.00}},

    'Side Dish': {
        'S1': {'name': 'Karkata Carbonara', 'price': 145.00},
        'S2': {'name': 'Puppy-paw Hashbrowns', 'price': 85.00},
        'S3': {'name': 'Forest Fresh Pizza', 'price': 350.00},
        'S4': {'name': 'Satisfying Salad', 'price': 235.00}},

    'Breakfast': {
        'B1': {'name': "Adventurer's Breakfast Sandwich", 'price': 65.00},
        'B2': {'name': 'Windward Waffles', 'price': 80.00},
        'B3': {'name': 'Stormcrest Pie', 'price': 50.00},
        'B4': {'name': 'Tantalizing TapSiLog', 'price': 75.00},
        'B5': {'name': 'TocSiLog Temptations', 'price': 75.00}},

    'Drinks': {
        'DR1': {'name': 'Distilled Water', 'price': 20.00},
        'DR2': {'name': 'Softdrinks', 'price': 25.00},
        'DR3': {'name': 'Wolfhook Juice', 'price': 55.00},
        'DR4': {'name': 'Fruits of the Fest', 'price': 69.00},
        'DR5': {'name': 'Osmanthus Burst', 'price': 120.00},
        'DR6': {'name': 'Dandelion Cider', 'price': 150.00}}
}
```

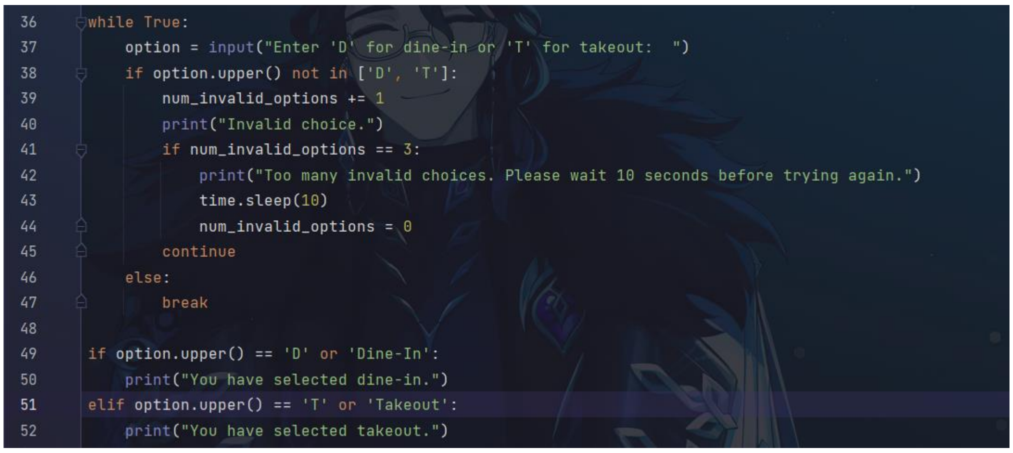
printreceipt.py

```
import random
from menu_dict import menu

business_name = "Traveler's Tavern"
order_number = random.randint(1000, 9999)
cashier = random.choice(['Diluc', 'Charles', 'Sara', 'Xiangling', 'Smiley
Yanxiao'])


def print_receipt(order_number, customer_name, cashier, dining_option,
payment_method, order_list, total_cost,
                  cash_received, change):
    with open(f"receipt_{order_number}.txt", "w") as f:
        f.write('---TRAVELER\'S TAVERN---\n')
        f.write('============================================\n')
        f.write('============================================\n')
        f.write(f'Order Number: {order_number}\n')
        f.write('============================================\n')
        f.write(f'Customer Name: {customer_name}\n')
        f.write(f'Cashier: {cashier}\n')
        f.write(f'Dining Option: {dining_option}\n')
        f.write(f'Payment Method: {payment_method}\n\n')
        f.write('Order List:\n')
        for item_code in order_list:
            item_code = item_code.upper()
            for category, items in menu.items():
                item_details = items.get(item_code)
                if item_details:
                    f.write(f"{item_details['name']} - PHP
{item_details['price']:.2f}\n")
                    break
            else:
                f.write(f"Invalid item code: {item_code}\n")
        f.write('\n')
        f.write(f'Total: PHP {total_cost:.2f}\n')
        f.write(f'Payment Method: {payment_method}\n')
        f.write(f'Received Cash: PHP {cash_received:.2f}\n')
        f.write(f'Change: PHP {change:.2f}\n\n')
        f.write('Thank you for dining with us!')
    print(f"Receipt {order_number} has been printed.")
```



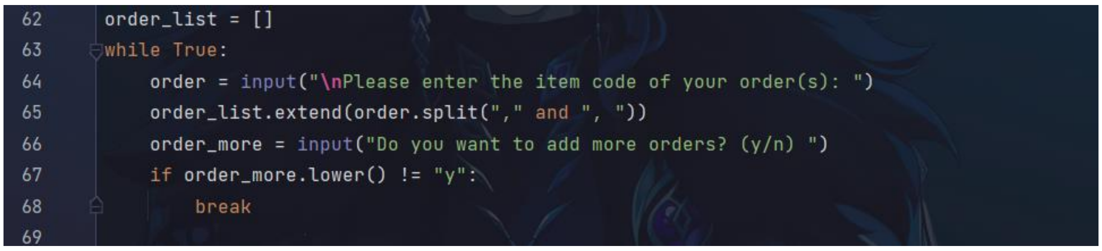
```
# Start the program by displaying a welcome message and asking the customer to enter their name.
print("Ad Astra Abyssosque!✨ Welcome to Traveler's Tavern!")
customerName = str(input("Customer Name:"))
print("Hello, " + customerName + "! Are you dining in or taking out?")
```

Figure 1. The program starts by displaying a welcome message and asking for the customer's name. This is an example of a **sequential structure** as instructions are executed one after another without any branching or looping.



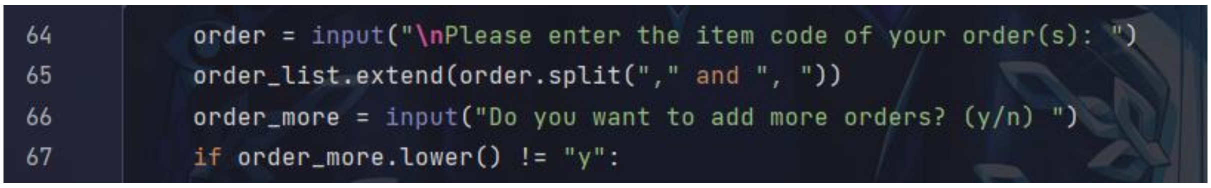
```python
while True:
    option = input("Enter 'D' for dine-in or 'T' for takeout:  ")
    if option.upper() not in ['D', 'T']:
        num_invalid_options += 1
        print("Invalid choice.")
        if num_invalid_options == 3:
            print("Too many invalid choices. Please wait 10 seconds before trying again.")
            time.sleep(10)
            num_invalid_options = 0
        continue
    else:
        break

if option.upper() == 'D' or 'Dine-In':
    print("You have selected dine-in.")
elif option.upper() == 'T' or 'Takeout':
    print("You have selected takeout.")
```

Figure 2. As shown above, the program contains **decision structures**, where the flow of execution is based on the value of a condition. For example, the program asks if the customer is dining in or taking out, and then checks if the input is valid.



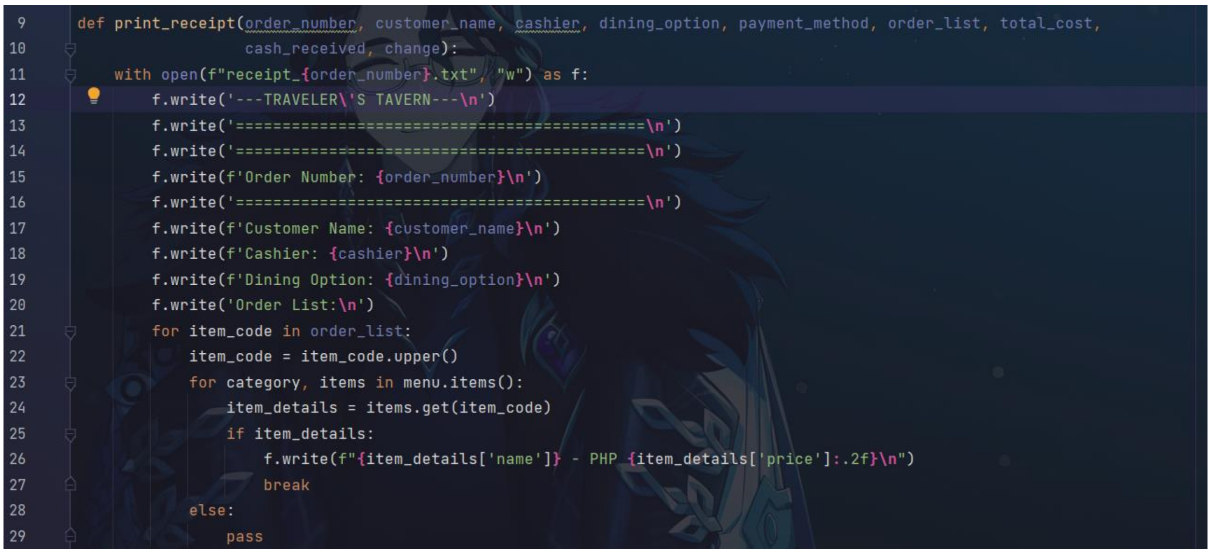
```python
order_list = []
while True:
    order = input("\nPlease enter the item code of your order(s): ")
    order_list.extend(order.split("," and ", "))
    order_more = input("Do you want to add more orders? (y/n) ")
    if order_more.lower() != "y":
        break

```

Figure 3. The program contains a **repetition structure**, where a block of code is repeated until a condition is met. For example, the program asks for the item code of the customer's order(s), and then asks if they want to add more orders.



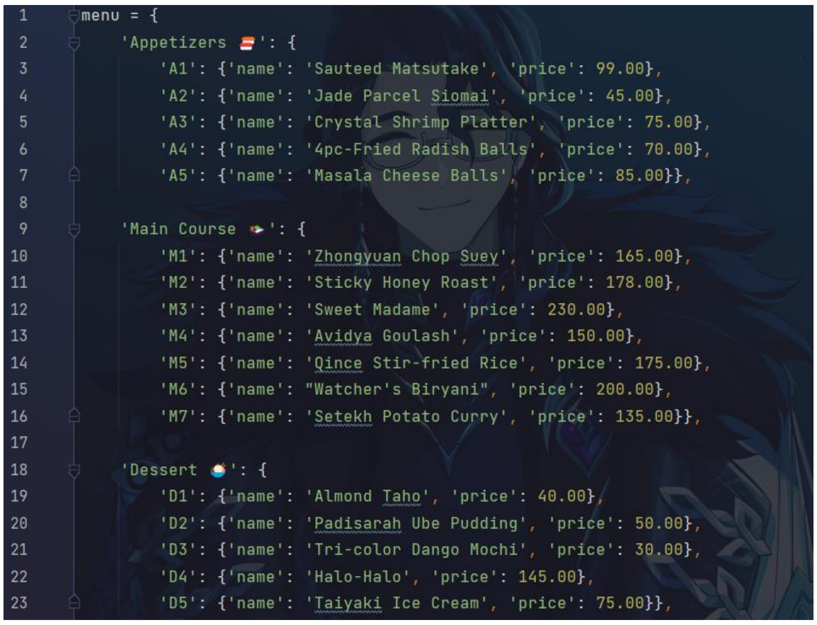
```python
    order = input("\nPlease enter the item code of your order(s): ")
    order_list.extend(order.split("," and ", "))
    order_more = input("Do you want to add more orders? (y/n) ")
    if order_more.lower() != "y":
```

Figure 4. The program uses some **string methods**, such as the lower () method to convert user inputs to lowercase and the split() method to split a string into a list**.**



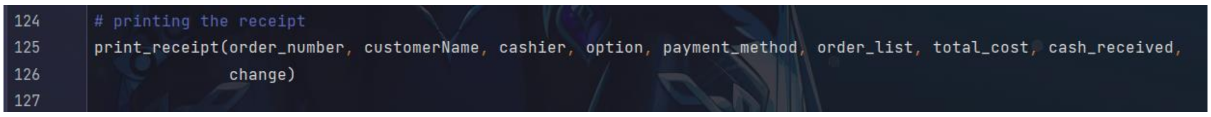
```python
def print_receipt(order_number, customer_name, cashier, dining_option, payment_method, order_list, total_cost,
                  cash_received, change):
    with open(f"receipt_{order_number}.txt", "w") as f:
        f.write('---TRAVELER\'S TAVERN---\n')
        f.write('============================================\n')
        f.write('============================================\n')
        f.write(f'Order Number: {order_number}\n')
        f.write('============================================\n')
        f.write(f'Customer Name: {customer_name}\n')
        f.write(f'Cashier: {cashier}\n')
        f.write(f'Dining Option: {dining_option}\n')
        f.write('Order List:\n')
        for item_code in order_list:
            item_code = item_code.upper()
            for category, items in menu.items():
                item_details = items.get(item_code)
                if item_details:
                    f.write(f"{item_details['name']} - PHP {item_details['price']:.2f}\n")
                    break
            else:
                pass
```

Figure 5. The code performs **text file manipulation** by creating a new text file and writing formatted data into it using the `open ()` and `write()` functions in Python. The function `print_receipt ()` writes the details of a customer's order into a text file named `receipt_{order_number}.txt` and prints a confirmation message to the console.

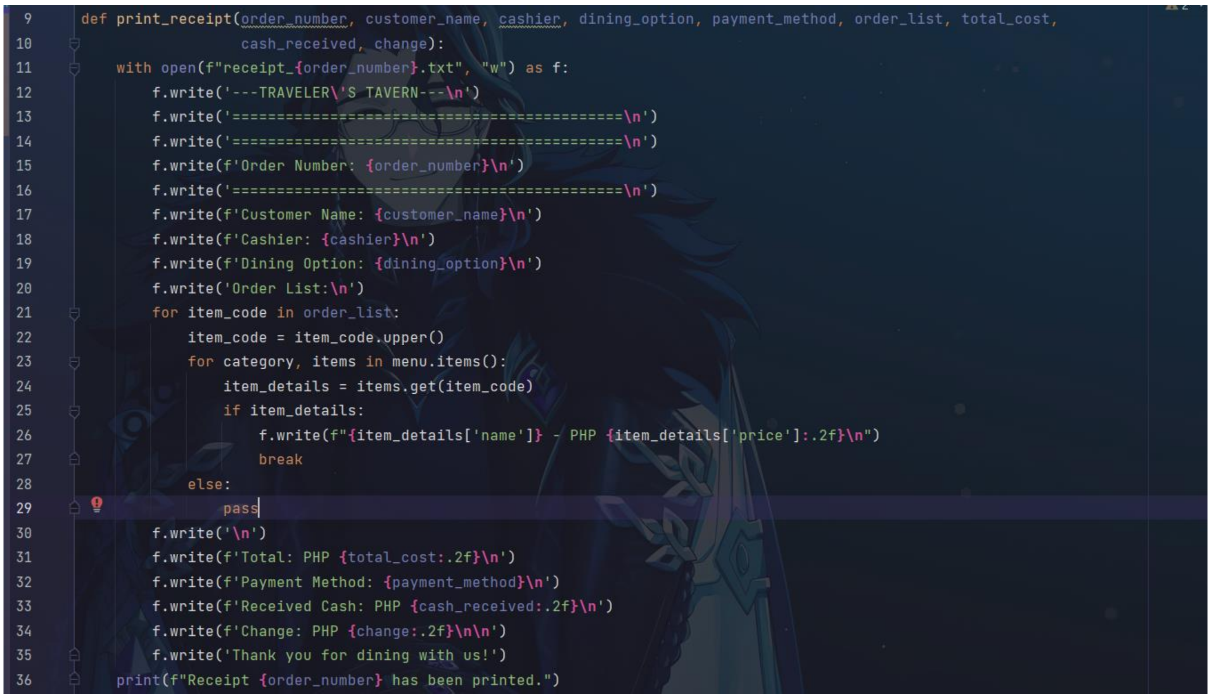


```python
menu = {
    'Appetizers 🍣': {
        'A1': {'name': 'Sauteed Matsutake', 'price': 99.00},
        'A2': {'name': 'Jade Parcel Siomai', 'price': 45.00},
        'A3': {'name': 'Crystal Shrimp Platter', 'price': 75.00},
        'A4': {'name': '4pc-Fried Radish Balls', 'price': 70.00},
        'A5': {'name': 'Masala Cheese Balls', 'price': 85.00}},

    'Main Course 🥙': {
        'M1': {'name': 'Zhongyuan Chop Suey', 'price': 165.00},
        'M2': {'name': 'Sticky Honey Roast', 'price': 178.00},
        'M3': {'name': 'Sweet Madame', 'price': 230.00},
        'M4': {'name': 'Avidya Goulash', 'price': 150.00},
        'M5': {'name': 'Qince Stir-fried Rice', 'price': 175.00},
        'M6': {'name': "Watcher's Biryani", 'price': 200.00},
        'M7': {'name': 'Setekh Potato Curry', 'price': 135.00}},

    'Dessert 🍧': {
        'D1': {'name': 'Almond Taho', 'price': 40.00},
        'D2': {'name': 'Padisarah Ube Pudding', 'price': 50.00},
        'D3': {'name': 'Tri-color Dango Mochi', 'price': 30.00},
        'D4': {'name': 'Halo-Halo', 'price': 145.00},
        'D5': {'name': 'Taiyaki Ice Cream', 'price': 75.00}},
```

Figure 6. As seen in the picture, the program uses **lists and dictionaries** to store and retrieve menu items. The menu is stored in a dictionary of dictionaries, where each category is a key, and its value is a dictionary of items, where the item code is the key, and its details are the value**.**

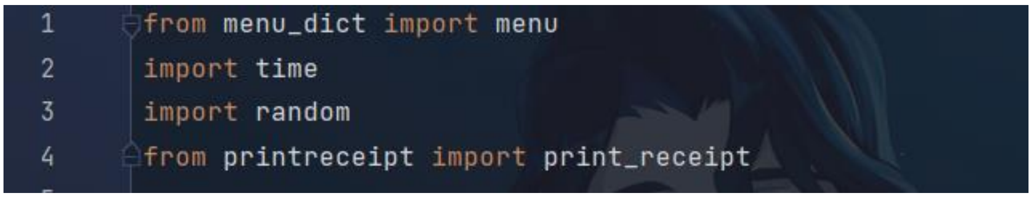


```python
# printing the receipt
print_receipt(order_number, customerName, cashier, option, payment_method, order_list, total_cost, cash_received,
              change)
```



```python
def print_receipt(order_number, customer_name, cashier, dining_option, payment_method, order_list, total_cost,
                  cash_received, change):
    with open(f"receipt_{order_number}.txt", "w") as f:
        f.write('---TRAVELER\'S TAVERN---\n')
        f.write('============================================\n')
        f.write('============================================\n')
        f.write(f'Order Number: {order_number}\n')
        f.write('============================================\n')
        f.write(f'Customer Name: {customer_name}\n')
        f.write(f'Cashier: {cashier}\n')
        f.write(f'Dining Option: {dining_option}\n')
        f.write('Order List:\n')
        for item_code in order_list:
            item_code = item_code.upper()
            for category, items in menu.items():
                item_details = items.get(item_code)
                if item_details:
                    f.write(f"{item_details['name']} - PHP {item_details['price']:.2f}\n")
                    break
            else:
                pass
        f.write('\n')
        f.write(f'Total: PHP {total_cost:.2f}\n')
        f.write(f'Payment Method: {payment_method}\n')
        f.write(f'Received Cash: PHP {cash_received:.2f}\n')
        f.write(f'Change: PHP {change:.2f}\n\n')
        f.write('Thank you for dining with us!')
    print(f"Receipt {order_number} has been printed.")
```

Figure 7. As shown above, the program uses a **function** called print_receipt () from the printreceipt module to print the receipt.



```python
from menu_dict import menu
import time
import random
from printreceipt import print_receipt
```

Figure 8. The program is **modularized** by importing modules, such as menu_dict and printreceipt, to separate functionalities and make the code more readable and organized.

CONCLUSIONS AND REALIZATIONS

When we first start coding, we felt intimidated and overwhelmed by the volume of information and new skills we needed to acquire. Despite our initial skepticism, we began to appreciate how coding could be an asset to our civil engineering program. Coding taught us to develop and use software to analyze complex structures and systems, automate repetitive tasks, and perform simulations. Above all, we learned the importance of problem-solving, which enabled us to tackle complex coding challenges by breaking them down into smaller, more manageable tasks.

Another significant lesson we learned from coding was the value of persistence. It's no secret that coding can be frustrating, especially when errors and bugs seem to pop up out of nowhere. However, through trial and error, we discovered that perseverance is key to overcoming these challenges. We learned that persistence and determination are essential traits in any field and that they can lead to great accomplishments.

Another significant lesson we learned from coding was the value of persistence. It's no secret that coding can be frustrating, especially when errors and bugs seem to pop up out of nowhere. However, through trial and error, we discovered that perseverance is key to overcoming these challenges. We learned that persistence and determination are essential traits in any field and that they can lead to great accomplishments.

Finally, we learned that coding is a never-ending journey of learning. While we acquired new skills and knowledge, we recognized that there is always more to learn. The skills we acquired through coding are transferable and can be applied in our daily lives, enabling us to become more proficient problem solvers and critical thinkers. Through coding, we gained a valuable set of skills that we will continue to hone and apply throughout our careers.

REFERENCES

Blackboard. (n.d.). *Blackboard Learn*. © 1997-2023 Blackboard Inc. All Rights Reserved. U.S. Patent No. 7,493,396 and 7,558,853. Additional Patents Pending. https://malayanmindanao.blackboard.com/ultra/courses/_12238_1/outline/edit/document/_703839_1?courseId=_12238_1&view=content

How to Print Colored Text in Python - Studytonight. (n.d.). https://www.studytonight.com/python-howtos/how-to-print-colored-text-in-python

MindTap - Cengage Learning. (n.d.-a). https://ng.cengage.com/static/nb/ui/evo/index.html?snapshotId=3392422&id=1747271202&deploymentId=6033602453271976910681101403&eISBN=9780357505397

MindTap - Cengage Learning. (n.d.-b). https://ng.cengage.com/static/nb/ui/evo/index.html?deploymentId=6033602453271976910681101403&eISBN=9780357505397&id=1747271247&snapshotId=3392422&

Python - Modify Strings. (n.d.). https://www.w3schools.com/python/python_strings_modify.asp

Python Functions. (n.d.). https://www.w3schools.com/python/python_functions.asp

Python Strings. (n.d.). https://www.w3schools.com/python/python_strings.asp