ASSIGNMENT: 3.4

Name: K.Abhinay

Roll No: 2303A51899

Batch :09

## DOCUMENTATION

Q. A retail store wants to automate its daily billing process.as a programmer, you are asked to develop a console-based application that assists the cashier in generating customer bills. generate python code the program should accept the customer name, the number of items purchased, and the price of each item. it should calculate the total bill amount and apply discount based on the total purchase value(for example, higher discounts for higher bill amounts).A The program must also calculate tax on the discounts amount and display the final payable bill clearly. The systems should allow the cashier to generate bills for multiple customers until to stop the program.

## OPTIMIZED CODE:

```python
#Retail Store billing system
#console based application to assist cashier in generating bills for
customers
# Fixed item catalog with item names and prices
# Fixed item catalog with prices and units
ITEMS_CATALOG = {
    1: {"name": "Rice (1kg)", "price": 50},
    2: {"name": "Wheat (1kg)", "price": 45},
    3: {"name": "Sugar (1kg)", "price": 55},
    4: {"name": "Salt (1kg)", "price": 20},
    5: {"name": "Oil (1L)", "price": 120},
    6: {"name": "Soap (1pack)", "price": 25},
    7: {"name": "Detergent (500g)", "price": 80},
    8: {"name": "Tea (250g)", "price": 180},
    9: {"name": "Coffee (250g)", "price": 200},
    10: {"name": "Milk Powder (400g)", "price": 250},
    11: {"name": "Flour (1kg)", "price": 40},
    12: {"name": "Lentils (1kg)", "price": 90},
    13: {"name": "Chickpeas (1kg)", "price": 85},
    14: {"name": "Butter (200g)", "price": 180},
    15: {"name": "Honey (500ml)", "price": 350},
    16: {"name": "Peanut Butter (500g)", "price": 220},
    17: {"name": "Pasta (500g)", "price": 60},
    18: {"name": "Noodles (6pack)", "price": 90},
    19: {"name": "Biscuits (200g)", "price": 40},
    20: {"name": "Chocolate (100g)", "price": 60},
    21: {"name": "Shampoo (250ml)", "price": 120},
    22: {"name": "Toothpaste (100ml)", "price": 50},
    23: {"name": "Hand Wash (250ml)", "price": 80},
    24: {"name": "Spices Mix (100g)", "price": 150},
}


def display_items():
    """Display all available items."""
```

```python
    print("\n" + "="*50)
    print("AVAILABLE ITEMS")
    print("="*50)
    for item_id, details in ITEMS_CATALOG.items():
        print(f"{item_id:2d}. {details['name']:<25}
₹{details['price']:.2f}")
    print("="*50)


def calculate_discount(total_amount):
    """Calculate discount based on total amount."""
    if total_amount > 5000:
        return total_amount * 0.15  # 15% discount
    elif total_amount > 3000:
        return total_amount * 0.10  # 10% discount
    elif total_amount > 1000:
        return total_amount * 0.08  # 8% discount
    elif total_amount > 500:
        return total_amount * 0.05  # 5% discount
    return 0  # No discount


def calculate_tax(total_amount):
    """Calculate 5% tax on total amount."""
    return total_amount * 0.05


def main():
    """Main function to run the billing system."""
    while True:
        customer_name = input("\nEnter customer name (or type 'exit' to
stop): ").strip()
        if customer_name.lower() == 'exit':
            print("Thank you for using our billing system!")
            break

        display_items()

        cart = {}
        total_amount = 0
```

```python
        while True:
            try:
                item_choice = input("\nEnter item number to add (or 'done'
to finish): ").strip()

                if item_choice.lower() == 'done':
                    if not cart:
                        print("No items selected. Please select at least
one item.")
                        continue
                    break

                item_id = int(item_choice)

                if item_id not in ITEMS_CATALOG:
                    print("Invalid item number. Please try again.")
                    continue

                quantity = int(input(f"Enter quantity for
{ITEMS_CATALOG[item_id]['name']}: "))

                if quantity <= 0:
                    print("Quantity must be greater than 0.")
                    continue

                item_price = ITEMS_CATALOG[item_id]["price"] * quantity

                if item_id in cart:
                    cart[item_id]["quantity"] += quantity
                    cart[item_id]["subtotal"] =
ITEMS_CATALOG[item_id]["price"] * cart[item_id]["quantity"]
                else:
                    cart[item_id] = {
                        "name": ITEMS_CATALOG[item_id]["name"],
                        "price": ITEMS_CATALOG[item_id]["price"],
                        "quantity": quantity,
                        "subtotal": item_price
                    }
```

```python
                total_amount += item_price
                print(f"Added {quantity} x
{ITEMS_CATALOG[item_id]['name']} - ₹{item_price:.2f}")

            except ValueError:
                print("Invalid input. Please enter a valid number or
'done'.")

        # Calculate discount and tax
        discount = calculate_discount(total_amount)
        tax = calculate_tax(total_amount - discount)
        final_amount = total_amount - discount + tax

        # Display receipt
        print("\n" + "="*60)
        print("RECEIPT")
        print("="*60)
        print(f"Customer Name: {customer_name}")
        print("-"*60)
        print("Item Details:")
        print("-"*60)

        for item_id, item_data in cart.items():
            print(f"{item_data['name']:<30} {item_data['quantity']:>5} x
₹{item_data['price']:>8.2f} = ₹{item_data['subtotal']:>10.2f}")

        print("-"*60)
        print(f"{'Subtotal':<45} ₹{total_amount:>12.2f}")
        print(f"{'Discount':<45} ₹{discount:>12.2f}")
        print(f"{'Tax (5%)':<45} ₹{tax:>12.2f}")
        print("="*60)
        print(f"{'FINAL AMOUNT':<45} ₹{final_amount:>12.2f}")
        print("="*60 + "\n")


if __name__ == '__main__':
    main()
```

**OUTPUT:**

Enter customer name (or type 'exit' to stop): K.Abhinay

```
==================================================
AVAILABLE ITEMS
==================================================
 1. Rice (1kg)          ₹50.00
 2. Wheat (1kg)         ₹45.00
 3. Sugar (1kg)         ₹55.00
 4. Salt (1kg)          ₹20.00
 5. Oil (1L)            ₹120.00
 6. Soap (1pack)        ₹25.00
 7. Detergent (500g)    ₹80.00
 8. Tea (250g)          ₹180.00
 9. Coffee (250g)       ₹200.00
10. Milk Powder (400g)  ₹250.00
11. Flour (1kg)         ₹40.00
12. Lentils (1kg)       ₹90.00
13. Chickpeas (1kg)     ₹85.00
14. Butter (200g)       ₹180.00
15. Honey (500ml)       ₹350.00
16. Peanut Butter (500g)  ₹220.00
17. Pasta (500g)        ₹60.00
18. Noodles (6pack)     ₹90.00
19. Biscuits (200g)     ₹40.00
20. Chocolate (100g)    ₹60.00
21. Shampoo (250ml)     ₹120.00
22. Toothpaste (100ml)  ₹50.00
23. Hand Wash (250ml)   ₹80.00
24. Spices Mix (100g)   ₹150.00
==================================================
```

Enter item number to add (or 'done' to finish): 5
Enter quantity for Oil (1L): 5
Added 5 x Oil (1L) - ₹600.00

Enter item number to add (or 'done' to finish): 4
Enter quantity for Salt (1kg): 3
Added 3 x Salt (1kg) - ₹60.00

Enter item number to add (or 'done' to finish): 6
Enter quantity for Soap (1pack): 2
Added 2 x Soap (1pack) - ₹50.00

Enter item number to add (or 'done' to finish): 8
Enter quantity for Tea (250g): 4
Added 4 x Tea (250g) - ₹720.00

Enter item number to add (or 'done' to finish): 9
Enter quantity for Coffee (250g): 12
Added 12 x Coffee (250g) - ₹2400.00

Enter item number to add (or 'done' to finish): done

```
============================================================
RECEIPT
============================================================
Customer Name: K.Abhinay
------------------------------------------------------------
Item Details:
------------------------------------------------------------
Oil (1L)              5 x ₹  120.00 = ₹    600.00
Salt (1kg)            3 x ₹   20.00 = ₹     60.00
Soap (1pack)          2 x ₹   25.00 = ₹     50.00
Tea (250g)            4 x ₹  180.00 = ₹    720.00
Coffee (250g)        12 x ₹  200.00 = ₹   2400.00
------------------------------------------------------------
Subtotal                      ₹     3830.00
Discount                      ₹      383.00
Tax (5%)                      ₹      172.35
============================================================
FINAL AMOUNT                  ₹     3619.35
============================================================
```

# ASSIGMENT_3.4

Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques Week2

Task 1: Zero-shot Prompt – Fibonacci Series Generator

Task Description #1

• Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers.

Expected Output #1

• A complete Python function generated by Copilot without any example provided.

• Correct output for sample input N = 7 → 0 1 1 2 3 5 8

• Observation on how Copilot understood the instruction with zero context.

Task 2: One-shot Prompt – List Reversal Function

Task Description #2

• Write a comment prompt to reverse a list and provide one

example below the comment to guide Copilot.

Expected Output #2

• Copilot-generated function to reverse a list using slicing or loop.

• Output: [3, 2, 1] for input [1, 2, 3]

• Observation on how adding a single example improved Copilot's accuracy.

Task 3: Few-shot Prompt – String Pattern Matching

Task Description #3

• Write a comment with 2–3 examples to help Copilot understand how to check if a string starts with a capital letter and ends with a period.

Expected Output #3

• A function is_valid() that checks the pattern.

• Output: True or False based on input.

• Students reflect on how multiple examples guide Copilot to generate more accurate code.

Task 4: Zero-shot vs Few-shot – Email Validator

Task Description #4

• First, prompt Copilot to write an email validation function using zero-shot (just the task in comment).

• Then, rewrite the prompt using few-shot examples.

Expected Output #4

• Compare both outputs:

Zero-shot may result in basic or generic validation.

Few-shot gives detailed and specific logic (e.g., @ and domain checking).

• Submit both code versions and note how few-shot improves reliability.

Task 5: Prompt Tuning – Summing Digits of a Number

Task Description #5

• Experiment with 2 different prompt styles to generate a function that returns the sum of digits of a number.

Style 1: Generic task prompt

Style 2: Task + Input/Output example

Expected Output #5

• Two versions of the sum_of_digits() function.

• Example Output: sum_of_digits(123) → 6

• Short analysis: which prompt produced cleaner or more optimized code and why?

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

# OPTIMIZED CODE:

```python
from typing import List, Tuple, Generator
from functools import lru_cache


@lru_cache(maxsize=128)
def _fib(n: int) -> int:
    """Calculate nth Fibonacci number with memoization."""
    if n < 0:
        raise ValueError("Index must be non-negative")
    return n if n <= 1 else _fib(n - 1) + _fib(n - 2)


def fibonacci_generator(n: int) -> Generator[int, None, None]:
    """Generate first N Fibonacci numbers efficiently."""
    if n <= 0:
        return
    a, b = 0, 1
    yield a
    if n > 1:
        yield b
        for _ in range(n - 2):
            a, b = b, a + b
            yield b
```

```python
def fibonacci_zero_shot(n: int) -> None:
    """Print first N Fibonacci numbers."""
    if n < 0:
        raise ValueError("N must be non-negative")
    print(' '.join(map(str, fibonacci_generator(n))))

def reverse_list(lst: List) -> List:
    """Reverse list using Python slicing."""
    return lst[::-1]


def reverse_list_inplace(lst: List) -> List:
    """Reverse list in-place using two-pointer technique."""
    left, right = 0, len(lst) - 1
    while left < right:
        lst[left], lst[right] = lst[right], lst[left]
        left += 1
        right -= 1
    return lst

def is_valid_sentence(text: str) -> bool:
    """Check if text starts with capital letter and ends with period."""
    if not text:
        return False
    return text[0].isupper() and text[-1] == '.'


_SENTENCE_TESTS: List[Tuple[str, bool]] = [
    ('Hello world.', True),
    ('hello world.', False),
    ('Hello world', False),
    ('A.', True),
    ('', False),
]

def validate_email_basic(email: str) -> bool:
    """Basic email validation (Zero-shot)."""
    return '@' in email and '.' in email
```

```python
def validate_email(email: str) -> bool:
    """Comprehensive email validation (Few-shot)."""
    if not email or not isinstance(email, str):
        return False

    if email.count('@') != 1:
        return False

    local, domain = email.rsplit('@', 1)

    if not local or len(local) > 64 or not domain or len(domain) < 3:
        return False

    if '.' not in domain or domain.startswith('.') or
domain.endswith('.'):
        return False

    if domain.startswith('-') or domain.endswith('-'):
        return False

    valid_chars =
set('abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789._%-+')
    return all(c in valid_chars for c in local)


_EMAIL_TESTS: List[Tuple[str, bool, bool]] = [
    ('user@example.com', True, True),
    ('invalid.email@.com', True, False),
    ('user@domain', False, False),
    ('user@domain.co.uk', True, True),
    ('@example.com', True, False),
]

def sum_of_digits_style1(n: int) -> int:
    """Sum digits using generator expression (Pythonic)."""
    return sum(int(d) for d in str(abs(n)))
```

```python
def sum_of_digits_style2(n: int) -> int:
    """Sum digits using iterative approach."""
    total = 0
    n = abs(n)
    while n > 0:
        total += n % 10
        n //= 10
    return total


def sum_of_digits_optimized(n: int) -> int:
    """Sum digits using map function (most optimized)."""
    return sum(map(int, str(abs(n))))


_DIGIT_SUM_TESTS: List[Tuple[int, int]] = [
    (123, 6),
    (9876, 30),
    (0, 0),
    (555, 15),
    (1, 1),
    (-123, 6),
]


def run_all_tests() -> None:
    """Execute all test cases with validation."""

    print("=" * 80)
    print("TASK 1: ZERO-SHOT FIBONACCI GENERATOR")
    print("=" * 80)
    print("Output for N=7: ", end="")
    fibonacci_zero_shot(7)
    print()

    print("=" * 80)
    print("TASK 2: ONE-SHOT LIST REVERSAL")
    print("=" * 80)
    test_list = [1, 2, 3]
    print(f"Input: {test_list}")
    print(f"Slicing: {reverse_list(test_list)}")
```

```python
    print(f"Two-pointer: {reverse_list_inplace([1, 2, 3])}")
    print()


    print("=" * 80)
    print("TASK 3: FEW-SHOT STRING PATTERN MATCHING")
    print("=" * 80)
    print(f"{'Text':<20} {'Result':<8} {'Status':<8}")
    print("-" * 36)
    for text, expected in _SENTENCE_TESTS:
        result = is_valid_sentence(text)
        status = "✓" if result == expected else "✗"
        print(f"{text:<20} {str(result):<8} {status:<8}")
    print()


    print("=" * 80)
    print("TASK 4: EMAIL VALIDATION COMPARISON")
    print("=" * 80)
    print(f"{'Email':<25} {'Basic':<8} {'Full':<8} {'Status':<8}")
    print("-" * 50)
    for email, basic_exp, full_exp in _EMAIL_TESTS:
        basic = validate_email_basic(email)
        full = validate_email(email)
        status = "✓" if (basic == basic_exp and full == full_exp) else
"✗"
        print(f"{email:<25} {str(basic):<8} {str(full):<8} {status:<8}")
    print()


    print("=" * 80)
    print("TASK 5: SUM OF DIGITS")
    print("=" * 80)
    print(f"{'Number':<10} {'Style1':<10} {'Style2':<10} {'Optimized':<10}
{'Status':<8}")
    print("-" * 50)
    for num, expected in _DIGIT_SUM_TESTS:
        s1 = sum_of_digits_style1(num)
        s2 = sum_of_digits_style2(num)
        opt = sum_of_digits_optimized(num)
        status = "✓" if (s1 == expected and s2 == expected and opt ==
expected) else "✗"
        print(f"{num:<10} {s1:<10} {s2:<10} {opt:<10} {status:<8}")
```

```
    print()


if __name__ == "__main__":
    run_all_tests()
```

**OUTPUT:**

```
==============================================================================
=======
TASK 1: ZERO-SHOT FIBONACCI GENERATOR
==============================================================================
=======
Output for N=7: 0 1 1 2 3 5 8


==============================================================================
=======
TASK 2: ONE-SHOT LIST REVERSAL
==============================================================================
=======
Input: [1, 2, 3]
Slicing: [3, 2, 1]
Two-pointer: [3, 2, 1]


==============================================================================
=======
TASK 3: FEW-SHOT STRING PATTERN MATCHING
==============================================================================
=======
Text            Result  Status
-------------------------------------
Hello world.      True    ✓
hello world.      False   ✓
Hello world       False   ✓
A.                True    ✓
                  False   ✓


==============================================================================
=======
TASK 4: EMAIL VALIDATION COMPARISON
==============================================================================
=======
Email             Basic   Full    Status
```

```
--------------------------------------------------
user@example.com         True    True    ✓
invalid.email@.com       True    False   ✓
user@domain              False   False   ✓
user@domain.co.uk        True    True    ✓
@example.com             True    False   ✓
```

================================================================================
========
TASK 5: SUM OF DIGITS
================================================================================
========

```
Number    Style1    Style2    Optimized  Status
--------------------------------------------------
123       6         6         6          ✓
9876      30        30        30         ✓
0         0         0         0          ✓
555       15        15        15         ✓
1         1         1         1          ✓
-123      6         6         6          ✓
```

PS C:\Users\abhin> & C:/Users/abhin/AppData/Local/Python/pythoncore-3.14-64/python.exe
c:/Users/abhin/OneDrive/Desktop/Assignment.py

================================================================================
========
TASK 1: ZERO-SHOT FIBONACCI GENERATOR
================================================================================
========
Output for N=7: 0 1 1 2 3 5 8

================================================================================
========
TASK 2: ONE-SHOT LIST REVERSAL
================================================================================
========
Input: [1, 2, 3]
Slicing: [3, 2, 1]
Two-pointer: [3, 2, 1]

================================================================================
========
TASK 3: FEW-SHOT STRING PATTERN MATCHING
```

```
================================================================
========
Text              Result   Status
------------------------------------
Hello world.      True     ✓
hello world.      False    ✓
Hello world       False    ✓
A.                True     ✓
                  False    ✓


================================================================
========
TASK 4: EMAIL VALIDATION COMPARISON
================================================================
========
Email               Basic   Full    Status
---------------------------------------------------
user@example.com        True    True    ✓
invalid.email@.com      True    False   ✓
user@domain             False   False   ✓
user@domain.co.uk       True    True    ✓
@example.com            True    False   ✓


================================================================
========
TASK 5: SUM OF DIGITS
================================================================
========
Number    Style1    Style2    Optimized  Status
-------------------------------------------------
123       6         6         6          ✓
9876      30        30        30         ✓
0         0         0         0          ✓
555       15        15        15         ✓
1         1         1         1          ✓
-123      6         6         6          ✓
```