AI ASSISTED CODING

ASSIGNMENT-10.1

Name: shiva sai.B

HT NO.:2403A51257

Batch No.:11

TASK 1

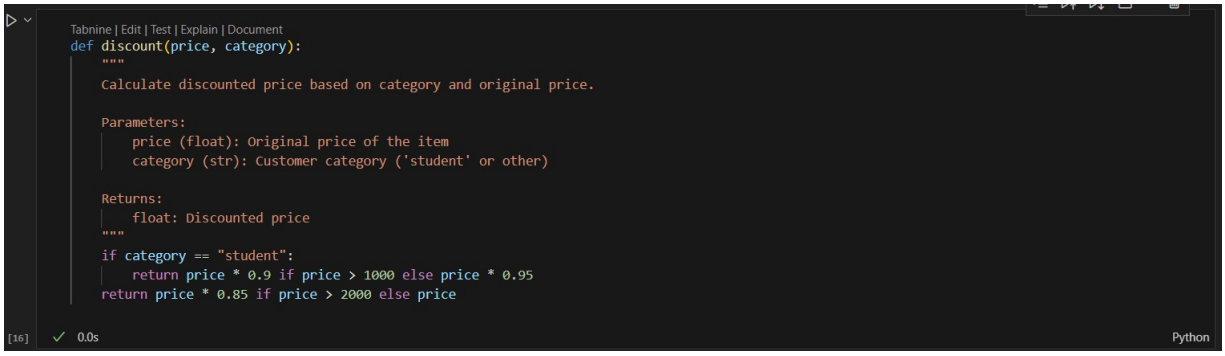
GIVEN CODE-

```
Tabnine | Edit | Test | Explain | Document
def discount(price, category):
    if category == "student":
        if price > 1000:
            return price * 0.9
        else:
            return price * 0.95
    else:
        if price > 2000:
            return price * 0.85
        else:
            return price
```

PROMPT-

For the above code simplify and refactor for readability.

AI GENERATED CODE-



```
Tabnine | Edit | Test | Explain | Document
def discount(price, category):
    """
    Calculate discounted price based on category and original price.

    Parameters:
        price (float): Original price of the item
        category (str): Customer category ('student' or other)

    Returns:
        float: Discounted price
    """
    if category == "student":
        return price * 0.9 if price > 1000 else price * 0.95
    return price * 0.85 if price > 2000 else price
[16]  ✓ 0.0s                                                    Python
```

TASK 2

GIVEN CODE-

```
Tabnine | Edit | Test | Explain | Document
def find_common(a, b):
    res = []
    for i in a:
        for j in b:
            if i == j:
                res.append(i)
    return res
```

PROMPT-

Refactor and optimize the above code.

AI GENERATED CODE-

```
Tabnine | Edit | Test | Explain | Document
def find_common(a, b):
    """
    Find common elements between two lists.

    Parameters:
        a (list): First list
        b (list): Second list

    Returns:
        list: List of common elements (no duplicates)
    """
    return list(set(a) & set(b))
```

✓ 0.0s Python

TASK 3

GIVEN CODE-

```
class Emp:
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, n, s):
        self.n = n
        self.s = s

    Tabnine | Edit | Test | Explain | Document
    def inc(self, p):
        self.s = self.s + (self.s * p / 100)

    Tabnine | Edit | Test | Explain | Document
    def pr(self):
        print("Emp:", self.n, "Salary:", self.s)
```

[20] ✓ 0.0s

PROMPT-

For the above code improve Naming conventions, Encapsulation, Readability C maintainability.

```python
class Employee:
    """
    Represents an employee with name and salary.

    Attributes:
        __name (str): Employee's name (private)
        __salary (float): Employee's salary (private)
    """
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, name: str, salary: float):
        self.__name = name
        self.__salary = salary
    Tabnine | Edit | Test | Explain | Document
    def increase_salary(self, percent: float):
        """
        Increase the employee's salary by a given percentage.

        Args:
            percent (float): Percentage increase (e.g., 10 for 10%)
        """
        if percent < 0:
            raise ValueError("Percentage cannot be negative")
        self.__salary += self.__salary * percent / 100
    Tabnine | Edit | Test | Explain | Document
    def get_salary(self) -> float:
        """Returns the current salary of the employee."""
        return self.__salary
    Tabnine | Edit | Test | Explain | Document
    def get_name(self) -> str:
        """Returns the name of the employee."""
        return self.__name
    Tabnine | Edit | Test | Explain | Document
    def display_info(self):
        """Prints the employee's name and salary."""
        print(f"Employee: {self.__name}, Salary: {self.__salary:.2f}")
```

TASK 4

GIVEN CODE-

```python
Tabnine | Edit | Test | Explain | Document
def process_scores(scores):
    total = 0
    for s in scores:
        total += s
    avg = total / len(scores)

    highest = scores[0]
    for s in scores:
        if s > highest:
            highest = s

    lowest = scores[0]
    for s in scores:
        if s < lowest:
            lowest = s

    print("Average:", avg)
    print("Highest:", highest)
    print("Lowest:", lowest)
```

PROMPT-

Break the code into smaller helper functions.

AI GENERATED CODE-

```
Tabnine | Edit | Test | Explain | Document
def calculate_average(scores):
    """Calculate and return the average of the scores."""
    return sum(scores) / len(scores)

Tabnine | Edit | Test | Explain | Document
def find_highest(scores):
    """Find and return the highest score."""
    highest = scores[0]
    for s in scores:
        if s > highest:
            highest = s
    return highest

Tabnine | Edit | Test | Explain | Document
def find_lowest(scores):
    """Find and return the lowest score."""
    lowest = scores[0]
    for s in scores:
        if s < lowest:
            lowest = s
    return lowest
Tabnine | Edit | Test | Explain | Document
def process_scores(scores):
    """
    Process a list of scores to print average, highest, and lowest values.
    """
    avg = calculate_average(scores)
    highest = find_highest(scores)
    lowest = find_lowest(scores)

    print(f"Average: {avg:.2f}")
    print(f"Highest: {highest}")
    print(f"Lowest: {lowest}")
# Example usage
scores = [78, 92, 85, 63, 99]
process_scores(scores)
```

OUTPUT-

```
Average: 83.40
Highest: 99
Lowest: 63
```

TASK 5

GIVEN CODE-

```
Tabnine | Edit | Test | Explain | Document
def div(a, b):
    return a / b

print(div(10, 0)) # This will raise a ZeroDivisionError
```

[24] ⊗ 1.4s Python

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[24], line 4
      1 def div(a, b):
      2     return a / b
----> 4 print(div(10, 0))  # Example with valid division

Cell In[24], line 2
      1 def div(a, b):
----> 2     return a / b

ZeroDivisionError: division by zero
```

PROMPT-

For the above code improve error handling, naming, and readability.

AI GENERATED CODE-

```
Tabnine | Edit | Test | Explain | Document
def divide(numerator, denominator):
    """
    Divide two numbers safely, handling division by zero.

    Parameters:
        numerator (float): The numerator
        denominator (float): The denominator

    Returns:
        float: The result of the division

    Raises:
        ValueError: If denominator is zero
    """
    if denominator == 0:
        raise ValueError("Denominator cannot be zero.")
    return numerator / denominator

# Example usage
try:
    result = divide(10, 0)
    print("Result:", result)
except ValueError as e:
    print("Error:", e)
```

[25] ✓ 0.0s Pytho

```
Error: Denominator cannot be zero.
```

TASK 6

GIVEN CODE-

```
Tabnine | Edit | Test | Explain | Document
def grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"

print(grade(85))
```

[26] ✓ 0.0s Python

```
B
```

PROMPT-

Simplify overly complex logic.

AI GENERATED CODE-

```
Tabnine | Edit | Test | Explain | Document
def grade(score):
    """
    Return the letter grade for a given numeric score.
    """
    grade_map = [(90, "A"), (80, "B"), (70, "C"), (60, "D")]

    for threshold, letter in grade_map:
        if score >= threshold:
            return letter
    return "F"

# Example usage
print(grade(85))  # Output: B
print(grade(72))  # Output: C
print(grade(58))  # Output: F
```

[27] ✓ 0.0s Python

```
B
C
F
```