

Assignment – 3.2

Task Description#1

Ask AI to write a function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example

CODE

```
def compound_interest(principal: float, rate: float, time: float, n: int) -> float:
```

```
    """
```

```
    Calculate compound interest.
```

```
    Args:
```

```
        principal (float): Initial amount invested or loaned.
```

```
        rate (float): Annual interest rate (in percentage).
```

```
        time (float): Time in years.
```

```
        n (int): Number of times interest is compounded per year.
```

```
    Returns:
```

```
        float: Final amount after compounding.
```

```
    """
```

```
    amount = principal * (1 + rate/(100*n))**(n*time)
```

```
    return amount
```

OUTPUT

```
1104.4861010000005
```

```
1938.8368221341054
```

Task Description#2

Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers.

CODE

```
import statistics
```

```
def calculate_stats(numbers: list) -> dict:
```

```

avg = statistics.mean(numbers)
med = statistics.median(numbers)
try:
    mod = statistics.mode(numbers)
except statistics.StatisticsError:
    mod = "No unique mode"
return {"average": avg, "median": med, "mode": mod}

nums = [1, 2, 2, 3, 4, 5, 5, 5, 6]
print(calculate_stats(nums))

nums2 = [10, 20, 30, 40]
print(calculate_stats(nums2))

```

OUTPUT

```

{'average': 3.6666666666666665, 'median': 4, 'mode': 5}
{'average': 25, 'median': 25.0, 'mode': 'No unique mode'}

```

Task Description#3

Provide multiple examples of input-output to the AI for `convert_to_binary(num)` function. Observe how AI uses few-shot prompting to generalize.

CODE

```

def convert_to_binary(num: int) -> str:
    if num == 0:
        return "0"
    binary = ""
    while num > 0:
        binary = str(num % 2) + binary
        num //= 2
    return binary

print(convert_to_binary(5)) # 101
print(convert_to_binary(8)) # 1000

```

```
print(convert_to_binary(15)) # 1111
print(convert_to_binary(0)) # 0
print(convert_to_binary(42)) # 101010
```

OUTPUT

```
101
1000
1111
0
101010
```

Task Description#4

Create an user interface for an hotel to generate bill based on customer requirements

CODE

```
MENU = {
    "Burger": 150,
    "Pizza": 300,
    "Pasta": 250,
    "Coffee": 50,
    "Tea": 30,
    "Sandwich": 120
}

def display_menu():
    print("\n--- Hotel Menu ---")
    for item, price in MENU.items():
        print(f"{item}: ₹{price}")
    print()

def take_order():
    order = {}
    while True:
```

```

    item = input("Enter item name (or 'done' to finish): ").title()

    if item == "Done":
        break

    if item not in MENU:
        print("Item not available. Try again.")
        continue

    qty = int(input(f"Enter quantity of {item}: "))

    if item in order:
        order[item] += qty
    else:
        order[item] = qty

    return order

def calculate_bill(order: dict) -> float:
    total = 0

    for item, qty in order.items():
        total += MENU[item] * qty

    return total

def generate_bill(order: dict):
    print("\n--- Your Bill ---")

    for item, qty in order.items():
        price = MENU[item] * qty
        print(f"{item} x {qty} = ₹{price}")

    total = calculate_bill(order)

    print(f"Total Amount: ₹{total}\n")

    print("Thank you for visiting our hotel!")

def main():
    display_menu()

    order = take_order()

```

```
    if order:
        generate_bill(order)
    else:
        print("No items ordered.")
if __name__ == "__main__":
    main()
```

OUTPUT

--- Hotel Menu ---

Burger: ₹150

Pizza: ₹300

Pasta: ₹250

Coffee: ₹50

Tea: ₹30

Sandwich: ₹120

Enter item name (or 'done' to finish): Burger

Enter quantity of Burger: 2

Enter item name (or 'done' to finish): Tea

Enter quantity of Tea: 3

Enter item name (or 'done' to finish): done

--- Your Bill ---

Burger x 2 = ₹300

Tea x 3 = ₹90

Total Amount: ₹390

Thank you for visiting our hotel!

Task Description#5

Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions

CODE

```
def convert_temperature(value: float, from_unit: str, to_unit: str) -> float:
```

```
    from_unit = from_unit.upper()
```

```
    to_unit = to_unit.upper()
```

```
    if from_unit == to_unit:
```

```
        return value
```

```
    # Convert from source to Celsius
```

```
    if from_unit == 'F':
```

```
        celsius = (value - 32) * 5/9
```

```
    elif from_unit == 'K':
```

```
        celsius = value - 273.15
```

```
    elif from_unit == 'C':
```

```
        celsius = value
```

```
    else:
```

```
        raise ValueError("Invalid from_unit")
```

```
    if to_unit == 'C':
```

```
        return celsius
```

```
    elif to_unit == 'F':
```

```
        return celsius * 9/5 + 32
```

```
    elif to_unit == 'K':
```

```
        return celsius + 273.15
```

```
    else:
```

```
        raise ValueError("Invalid to_unit")
```

```
# Example usage
```

```
print(convert_temperature(0, 'C', 'F')) # 32.0
```

```
print(convert_temperature(100, 'C', 'K')) # 373.15
```

```
print(convert_temperature(212, 'F', 'C')) # 100.0
```

OUTPUT

```
32.0
```

373.15

100.0