

L.Aravind Reddy

2303A52028

Batch - 38

Task 1: Progressive Prompting – Calculator Design

Prompt

Design a simple calculator function. Gradually refine the prompt to include arithmetic operations and error handling.

Explanation

As the prompt becomes more specific, the calculator logic improves by handling operators correctly and preventing division by zero.

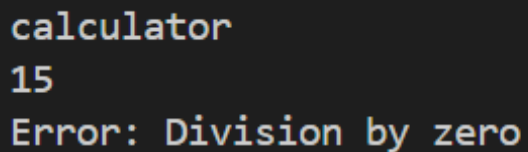
Code

```
def calculator(a, b, op):  
    if op == '+':  
        return a + b  
    elif op == '-':  
        return a - b  
    elif op == '*':  
        return a * b  
    elif op == '/':  
        if b == 0:  
            return "Error: Division by zero"  
        return a / b  
    else:  
        return "Invalid operator"
```

Output

calculator(10, 5, '+') → 15

calculator(10, 0, '/') → Error: Division by zero



```
calculator  
15  
Error: Division by zero
```

Task 2: Refining Prompts – Sorting Student Marks

Prompt

Sort student marks in ascending order without using built-in sorting functions.

Explanation

Clear prompt instructions lead to a correct comparison-based sorting algorithm that handles duplicate values.

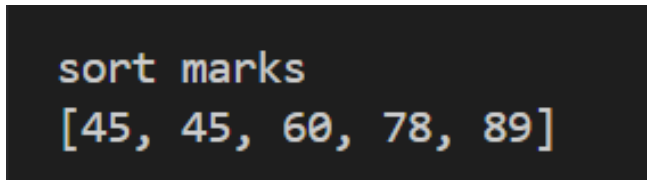
Code

```
def sort_marks(marks):  
    n = len(marks)  
    for i in range(n):  
        for j in range(i + 1, n):  
            if marks[i] > marks[j]:  
                marks[i], marks[j] = marks[j], marks[i]  
    return marks
```

Output

Input: [78, 45, 89, 45, 60]

Output: [45, 45, 60, 78, 89]

A terminal window with a dark background and light-colored text. The text shows the function call 'sort marks' followed by the output list '[45, 45, 60, 78, 89]' on the next line.

```
sort marks  
[45, 45, 60, 78, 89]
```

Task 3: Few-Shot Prompting – Prime Number Validation

Prompt

Check whether a number is prime using few-shot examples.

Explanation

Few-shot examples improve correctness and ensure edge cases like 0 and 1 are handled.

Code

```
def is_prime(n):  
    if n <= 1:  
        return False  
    for i in range(2, int(n ** 0.5) + 1):  
        if n % i == 0:  
            return False
```

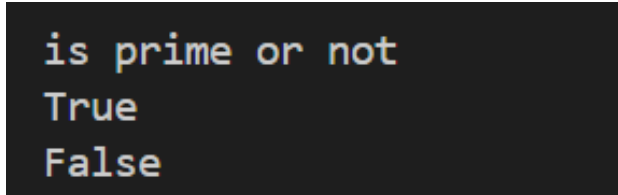
```
return True
```

Output

is_prime(2) → True

is_prime(4) → False

is_prime(1) → False



```
is prime or not
True
False
```

Task 4: Prompt-Guided UI – Student Grading System

Prompt

Create a user interface that accepts marks and calculates total, percentage, and grade.

Explanation

Prompt clarity results in a structured grading system with accurate calculations.

Code

```
def grading_system():
    m1 = int(input("Enter marks 1: "))
    m2 = int(input("Enter marks 2: "))
    m3 = int(input("Enter marks 3: "))

    total = m1 + m2 + m3
    percentage = total / 3

    if percentage >= 90:
        grade = 'A'
    elif percentage >= 75:
        grade = 'B'
    elif percentage >= 60:
        grade = 'C'
    else:
        grade = 'Fail'

    print("Total:", total)
    print("Percentage:", percentage)
    print("Grade:", grade)
```

Output

Input: 80, 75, 85

Output: Total = 240, Percentage = 80, Grade = B

```
grading system
240
80.0
B
```

Task 5: Prompt Specificity – Unit Conversion

Prompt

Convert kilometers to miles and miles to kilometers using clear instructions.

Explanation

Specific prompts improve accuracy and correctness of unit conversion logic.

Code

```
def convert_units(value, unit):
    if unit == 'km_to_miles':
        return value * 0.621371
    elif unit == 'miles_to_km':
        return value / 0.621371
    else:
        return "Invalid conversion type"
```

Output

convert_units(10, 'km_to_miles') → 6.21

```
convert units
6.21371
9.977935886933894
```