ASSIGNMENT 9

RAMIDI . SAI CHARAN

2403A52124

TASK 1 :

```python
def sum_even_odd_ai_docstring(numbers):
    even_sum = 0
    odd_sum = 0
    for number in numbers:
        if number % 2 == 0:
            even_sum += number
        else:
            odd_sum += number
    return even_sum, odd_sum

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum_ai, odd_sum_ai = sum_even_odd_ai_docstring(my_list)
print(f"Sum of even numbers (AI docstring version): {even_sum_ai}")
print(f"Sum of odd numbers (AI docstring version): {odd_sum_ai}")
```

```
Sum of even numbers (AI docstring version): 30
Sum of odd numbers (AI docstring version): 25
```

EXPLANATION :

This Python code defines a function called `sum_even_odd_ai_docstring` that takes one argument, a list named `numbers`.

Inside the function:

- It initializes two variables, `even_sum` and `odd_sum`, to 0. These will store the sums of even and odd numbers respectively.
- It then iterates through each `number` in the input `numbers` list.
- For each `number`, it checks if the number is even using the modulo operator (`%`). If `number % 2 == 0` is true, the number is even and is added to `even_sum`. Otherwise, the number is odd and is added to `odd_sum`.
- After the loop finishes, the function returns a tuple containing the final `even_sum` and `odd_sum`.

The code then demonstrates how to use this function:

- It creates a sample list `my_list`.
- It calls the `sum_even_odd_ai_docstring` function with `my_list` and unpacks the returned tuple into `even_sum_ai` and `odd_sum_ai`.
- Finally, it prints the calculated sum of even and odd numbers with descriptive labels.

The docstring within the function provides a brief explanation of what the function does, its arguments, and what it returns.

TASK 2 :

```python
class sru_student:
    # Constructor method to initialize the student attributes
    def __init__(self, name, roll_no, hostel_status):
        # Initialize the student's name
        self.name = name
        # Initialize the student's roll number
        self.roll_no = roll_no
        # Initialize the student's hostel status (True if in hostel, False otherwise)
        self.hostel_status = hostel_status
        # Initialize the student's fee, starting at 0
        self.fee = 0

    # Method to update the student's fee
    def fee_update(self, amount):
        # Add the specified amount to the current fee
        self.fee += amount
        # Print a confirmation message
        print(f"Fee updated for {self.name}. New fee: {self.fee}")

    # Method to display the student's details
    def display_details(self):
        # Print the student's name
        print(f"Name: {self.name}")
        # Print the student's roll number
        print(f"Roll No.: {self.roll_no}")
        # Print the student's hostel status
        print(f"Hostel Status: {self.hostel_status}")
        # Print the student's current fee
        print(f"Current Fee: {self.fee}")

# --- Example usage with user input ---

# Get student details from user input
name = input("Enter student's name: ")
roll_no = input("Enter student's roll number: ")
hostel_status_input = input("Is the student in a hostel? (yes/no): ").lower()
hostel_status = True if hostel_status_input == 'yes' else False

# Create a new student object with user-provided data
student1 = sru_student(name, roll_no, hostel_status)
```

```python
# Display the initial details of the student
print("\n--- Initial Student Details ---")
student1.display_details()

# Get fee update amount from user input
try:
    fee_amount = int(input("\nEnter the fee amount to add: "))
    student1.fee_update(fee_amount)
except ValueError:
    print("Invalid amount. Please enter a number.")

# Display the updated details of the student
print("\n--- Updated Student Details ---")
student1.display_details()
```

```
Enter student's name: SAI CHARAN
Enter student's roll number: 2403A52124
Is the student in a hostel? (yes/no): YES

--- Initial Student Details ---
Name: SAI CHARAN
Roll No.: 2403A52124
Hostel Status: True
Current Fee: 0

Enter the fee amount to add: 1000000
Fee updated for SAI CHARAN. New fee: 1000000

--- Updated Student Details ---
Name: SAI CHARAN
Roll No.: 2403A52124
Hostel Status: True
Current Fee: 1000000
```

EXPLANATION :

1. The `sru_student` Class

This part of the code defines the `sru_student` class, which acts as a blueprint for creating student objects.

- `__init__(self, name, roll_no, hostel_status)`: This is the **constructor method**. It runs automatically whenever a new `sru_student` object is created.
  - `self`: Refers to the specific object being created.
  - `name`, `roll_no`, `hostel_status`: These are the parameters that must be provided when you create a student. The method assigns these values to the object's attributes (e.g., `self.name = name`).
  - `self.fee = 0`: This line initializes the `fee` attribute for every new student to `0`.
- `fee_update(self, amount)`: This method is used to add to the student's fee. It takes an `amount` as an argument and adds it to the existing `self.fee`.
- `display_details(self)`: This method simply prints the current values of all the student's attributes in a clear, formatted way.

2. Example Usage with User Input

This section demonstrates how to use the `sru_student` class, but instead of using fixed values, it prompts the user to enter the information.

- **Getting Student Details**: The code uses the `input()` function to ask the user for the student's name, roll number, and hostel status. The `input()` function returns a string, so the code converts the hostel status input (`'yes'` or `'no'`) into a boolean value (`True` or `False`).
- **Creating the Object**: `student1 = sru_student(name, roll_no, hostel_status)` creates a new `sru_student` object using the details provided by the user.
- **Displaying Initial Details**: `student1.display_details()` is called to show the student's information before any fee updates.
- **Updating the Fee**: The code prompts the user to enter the fee amount to be added.
  - `try...except ValueError` : This is an **error-handling block**. It attempts to convert the user's input for the fee amount to an integer using `int()` .
  - If the user enters a non-numeric value (like "abc"), the `int()` function will raise a `ValueError` . The `except` block catches this error and prints a friendly message instead of crashing the program.
- **Displaying Updated Details**: Finally, `student1.display_details()` is called again to show the student's details, now including the updated fee.

TASK 3 :

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Error! Division by zero is not allowed."
    return a / b
try:
    num1 = float(input("Enter the first number: "))
    operator = input("Enter an operator (+, -, *, /): ")
    num2 = float(input("Enter the second number: "))
    if operator == '+':
        result = add(num1, num2)
    elif operator == '-':
        result = subtract(num1, num2)
    elif operator == '*':
        result = multiply(num1, num2)
    elif operator == '/':
        result = divide(num1, num2)
    else:
        result = "Invalid operator. Please use one of +, -, *, /."
    print(f"The result is: {result}")

except ValueError:
    print("Invalid input. Please enter valid numbers.")
```

```
Enter the first number: 25
Enter an operator (+, -, *, /): +
Enter the second number: 25
The result is: 50.0
```

EXPLANATION :

1. Function Definitions

The script begins by defining four functions:

- `add(a, b)` : Takes two arguments, `a` and `b` , and returns their sum.
- `subtract(a, b)` : Returns the difference between `a` and `b` .
- `multiply(a, b)` : Returns the product of `a` and `b` .
- `divide(a, b)` : This function is crucial because it includes a check for division by zero. If `b` is `0` , it returns an error message instead of raising a `ZeroDivisionError` which would crash the program. Otherwise, it returns the quotient of `a` divided by `b` .

2. User Input

The program then prompts the user for three pieces of information using the `input()` function:

- `num1` : The first number.
- `operator` : The mathematical operator (`+` , `-` , `*` , or `/`).
- `num2` : The second number.

A `try...except` block is used to handle potential errors. The `float()` function attempts to convert the user's number inputs from strings to floating-point numbers. If the user enters text that cannot be converted (e.g., "hello"), a `ValueError` is raised, and the `except` block catches it, printing an "Invalid input" message.

3. Conditional Logic

An `if/elif/else` block checks the value of the `operator` variable.

- If `operator` is `'+'` , the `add` function is called.
- If `operator` is `'-'` , the `subtract` function is called.
- If `operator` is `'*'` , the `multiply` function is called.
- If `operator` is `'/'` , the `divide` function is called.
- If the user enters any other character, the `else` block is executed, and an "Invalid operator" message is assigned to the `result` variable.

4. Output

Finally, the `print()` function displays the result of the calculation. The `f-string` is used to format the output, embedding the value of the `result` variable directly into the printed string.