23FOTCA13902

Q 1 - 5	Questions of (Tkinter, Class and object) using exception handling

Code

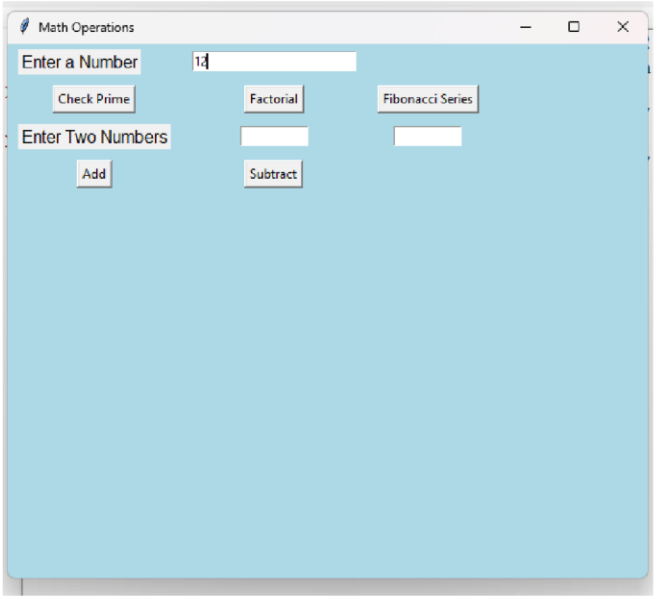
```
# Q 1 to 5

from tkinter import *
from tkinter import messagebox

t = Tk()
t.title("Math Operations")
t.geometry("600x500")
t.configure(bg="lightblue")

class MathOperations:     def
__init__(self, master):
self.master = master

        Label(master, text="Enter a Number", font=("Arial", 12)).grid(row=0,
column=0, padx=10, pady=5, sticky="w")
        self.num = StringVar()
        Entry(master, textvariable=self.num, width=25).grid(row=0, column=1,
padx=10, pady=5)

        Button(master, text="Check Prime", command=self.check_prime).grid(row=1,
column=0, padx=10, pady=5)
        Button(master, text="Factorial",
command=self.calculate_factorial).grid(row=1, column=1, padx=10, pady=5)
Button(master, text="Fibonacci Series",
command=self.fibonacci_series).grid(row=1, column=2, padx=10, pady=5)

        Label(master, text="Enter Two Numbers", font=("Arial", 12)).grid(row=2,
column=0, padx=10, pady=5, sticky="w")
        self.num1 = StringVar()
self.num2 = StringVar()
        Entry(master, textvariable=self.num1, width=10).grid(row=2, column=1,
padx=5, pady=5)
        Entry(master, textvariable=self.num2, width=10).grid(row=2, column=2,
padx=5, pady=5)

        Button(master,  text="Add",  command=self.addition).grid(row=3,  column=0,
padx=10, pady=5)
        Button(master, text="Subtract", command=self.subtraction).grid(row=3,
column=1, padx=10, pady=5)

    def check_prime(self):
try:
            num = int(self.num.get())
if num < 2:
```

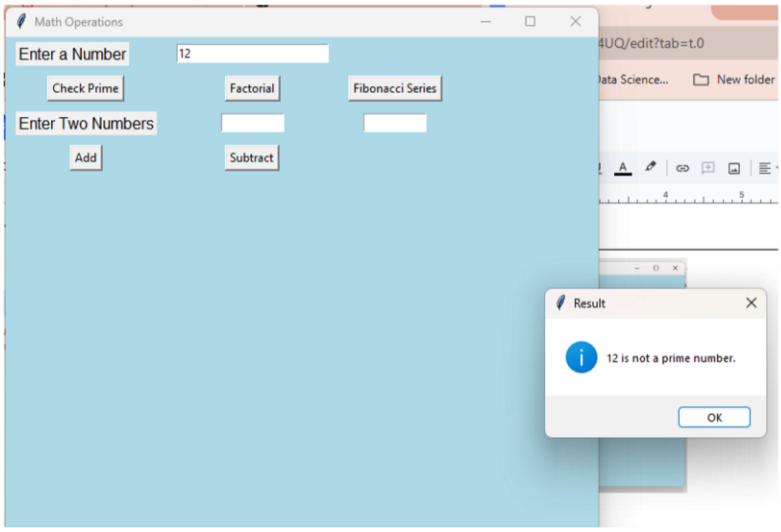
```
            messagebox.showinfo("Result", f"{num} is not a prime number.")
return
```

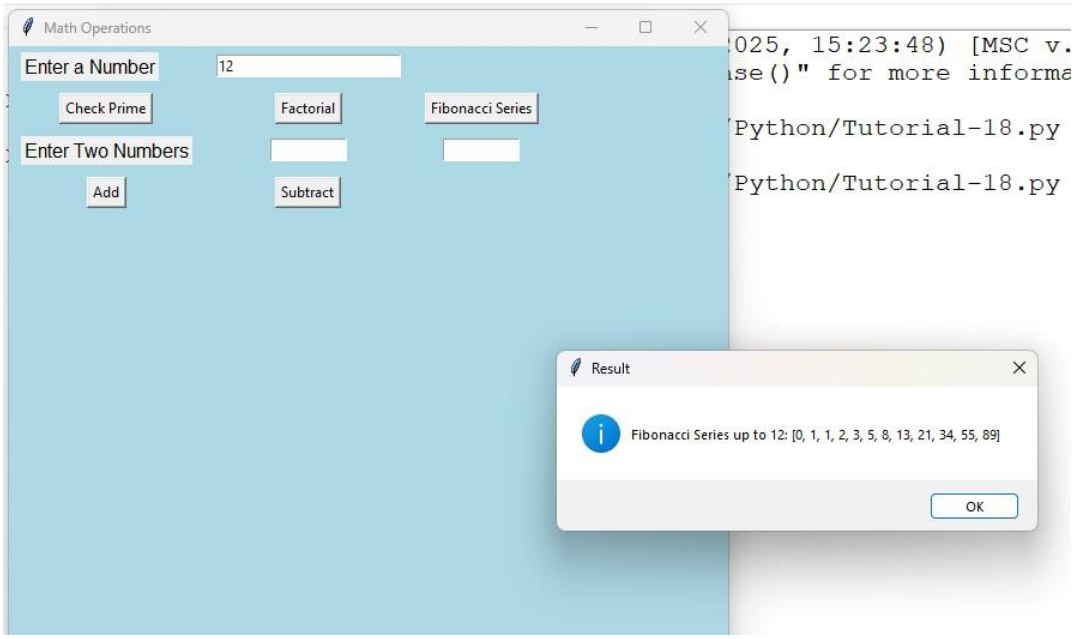
```
        for i in range(2, int(num ** 0.5) + 1):
if num % i == 0:
                messagebox.showinfo("Result", f"{num} is not a prime number.")
return
        messagebox.showinfo("Result", f"{num} is a prime number.")
except ValueError:          messagebox.showerror("Error", "Invalid Input!
Please enter a valid number.")

    def calculate_factorial(self):
try:
        num = int(self.num.get())
if num < 0:
            messagebox.showerror("Error", "Factorial is not defined for negative
numbers!")
            return          fact = 1
for i in range(1, num + 1):
fact *= i
        messagebox.showinfo("Result", f"Factorial of {num} is {fact}")
except ValueError:          messagebox.showerror("Error", "Invalid Input!
Please enter a valid number.")

    def fibonacci_series(self):
try:
        num = int(self.num.get())
if num < 0:
            messagebox.showerror("Error", "Please enter a non-negative number.")
return          series = [0, 1]          for _ in range(2, num):
series.append(series[-1] + series[-2])          messagebox.showinfo("Result",
f"Fibonacci Series up to {num}:
{series[:num]}")        except ValueError:
messagebox.showerror("Error", "Invalid Input! Please enter a valid
number.")

    def addition(self):
try:
        num1, num2 = int(self.num1.get()), int(self.num2.get())
result = num1 + num2          messagebox.showinfo("Result", f"Addition
Result: {result}")        except ValueError:
messagebox.showerror("Error", "Invalid Input! Please enter valid
numbers.")

    def subtraction(self):
try:
```

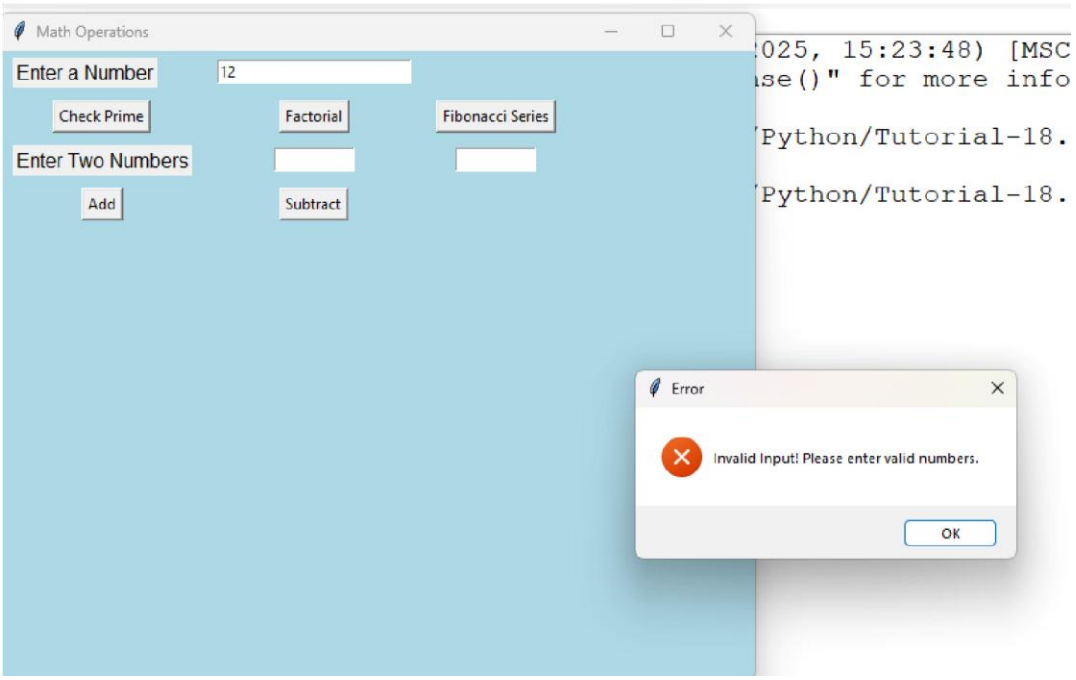
```
            num1, num2 = int(self.num1.get()), int(self.num2.get())
            result = num1 - num2
            messagebox.showinfo("Result", f"Subtraction Result: {result}")
        except ValueError:
            messagebox.showerror("Error", "Invalid Input! Please enter valid
numbers.")

app = MathOperations(t)
t.mainloop()
```

Output






Math Operations
Enter a Number
12
Check Prime
Factorial
Fibonacci Series
Enter Two Numbers
Add
Subtract
Result
Fibonacci Series up to 12: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
OK


Math Operations
Enter a Number
12
Check Prime
Factorial
Fibonacci Series
Enter Two Numbers
Add
Subtract
Error
Invalid Input! Please enter valid numbers.
OK

23FOTCA13902

2	6-10 : Console Programs

Code

```
#Q 6 to 10

def find_max():
try:
        a = float(input("Enter first number: "))         b =
float(input("Enter second number: "))         c =
float(input("Enter third number: "))         print(f"The
maximum number is: {max(a, b, c)}")     except
ValueError:
        print("Invalid input! Please enter numeric values.")

find_max()



def reverse_string():
try:
        s = input("Enter a string: ")
if not s:
            raise ValueError("String cannot be empty!")
        print(f"Reversed string: {s[::-1]}")
except ValueError as e:
        print(f"Error: {e}")

reverse_string()

def count_vowels_consonants():
try:
        s = input("Enter a string: ")
if not s:
            raise ValueError("String cannot be empty!")

        vowels = "aeiouAEIOU"
        vowel_count = sum(1 for char in s if char in vowels)
        consonant_count = sum(1 for char in s if char.isalpha() and char not in vowels)

        print(f"Vowels: {vowel_count}, Consonants: {consonant_count}")
except ValueError as e:
        print(f"Error: {e}")

count_vowels_consonants()



def check_even_odd():
try:
```

```
num = int(input("Enter a number: "))        print(f"{num} is
Even" if num % 2 == 0 else f"{num} is Odd")     except
ValueError:
```

```
        print("Invalid input! Please enter an integer.")

check_even_odd()


def is_leap_year():
try:
        year = int(input("Enter a year: "))        if (year % 4 == 0 and
year % 100 != 0) or (year % 400 == 0):
            print(f"{year} is a Leap Year.")
else:
            print(f"{year} is NOT a Leap Year.")
except ValueError:
        print("Invalid input! Please enter a valid year.")

is_leap_year()
```

Output

```
================================ RESTART: D:/P
Enter first number: 19
Enter second number: 34
Enter third number: 33
The maximum number is: 34.0
Enter a string: DUSHYANTSInh
Reversed string: hnISTNAYHSUD
Enter a string: 22FOTCA11025
Vowels: 2, Consonants: 3
Enter a number: 44
44 is Even
Enter a year: 2025
2025 is NOT a Leap Year.
>>>
```

23FOTCA13902

3	11-15 : Pattern (Integer)

Code

```
#11 to 15


def right_triangle(n):     try:
for i in range(1, n + 1):
            print("* " * i)     except ValueError:
print("Invalid input! Please enter a valid number.")

n = int(input("Enter the number of rows: "))
right_triangle(n)


def inverted_triangle(n):
try:         for i in range(n, 0,
-1):
            print("* " * i)     except ValueError:
print("Invalid input! Please enter a valid number.")

n = int(input("Enter the number of rows: "))
inverted_triangle(n)


def number_pyramid(n):     try:         for i in range(1, n +
1):             print(" ".join(str(j) for j in range(1, i + 1)))
except ValueError:         print("Invalid input! Please
enter a valid number.")

n = int(input("Enter the number of rows: "))
number_pyramid(n)


def diamond_pattern(n):     try:
for i in range(1, n + 1, 2):
            print(" " * ((n - i) // 2) + "* " * i)
for i in range(n - 2, 0, -2):
            print(" " * ((n - i) // 2) + "* " * i)     except
ValueError:         print("Invalid input! Please enter a
valid number.")

n = int(input("Enter an odd number for diamond shape: "))
diamond_pattern(n)
```

```
from math import factorial

def pascal_triangle(n):     try:
for i in range(n):            for j in
range(n - i + 1):
                print(end=" ")  # Space padding            for j in range(i
+ 1):                print(factorial(i) // (factorial(j) * factorial(i - j)),
end=" ")              print()     except ValueError:
        print("Invalid input! Please enter a valid number.")

n = int(input("Enter the number of rows: "))
pascal_triangle(n)
```

Output

```
================================ RESTART: D:/Python/Tutorial-18.py =====
Enter the number of rows: 4
*
* *
* * *
* * * *
Enter the number of rows: 5
* * * * *
* * * *
* * *
* *
*
Enter the number of rows: 6
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
Enter an odd number for diamond shape: 4
 *
* * *
 * *
Enter the number of rows: 3
    1
   1 1
  1 2 1
>>>
```

4	16-20 :Pattern (String)

Code

```
#16 to 20

def alphabet_triangle(n):     try:         for i in range(1, n
+ 1):             print(" ".join(chr(65 + j) for j in range(i)))
except ValueError:         print("Invalid input! Please
enter a valid number.")

n = int(input("Enter the number of rows: "))
alphabet_triangle(n)



def inverted_alphabet_triangle(n):     try:         for i in
range(n, 0, -1):             print(" ".join(chr(65 + j) for j in
range(i)))     except ValueError:         print("Invalid
input! Please enter a valid number.")

n = int(input("Enter the number of rows: "))
inverted_alphabet_triangle(n)



def diamond_alphabet_pattern(n):     try:         for i in range(n):
print(" " * (n - i - 1) + " ".join(chr(65 + j) for j in range(i + 1)))
for i in range(n - 2, -1, -1):             print(" " * (n - i - 1) + "
".join(chr(65 + j) for j in range(i + 1)))     except ValueError:
print("Invalid input! Please enter a valid number.")

n = int(input("Enter the number of rows: "))
diamond_alphabet_pattern(n)



from math import factorial

def pascal_alphabet_triangle(n):
try:         for i in range(n):
for j in range(n - i + 1):
                print(end=" ")  # Space padding             for j in range(i + 1):
print(chr(65 + (factorial(i) // (factorial(j) * factorial(i - j)) - 1)), end=" ")
```

	print() except ValueError: print("Invalid input! Please enter a valid number.") n = int(input("Enter the number of rows: ")) pascal_alphabet_triangle(n)
Output	========== RESTART: D:/Python/Tutorial-18.py ====== Enter the number of rows: 3 A A B A B C Enter the number of rows: 5 A B C D E A B C D A B C A B A Enter the number of rows: 5 A A B A B C A B C D A B C D E A B C D A B C A B A Enter the number of rows: 5 A A A A B A A C C A A D F D A >>>

5	21-25 : List, Set, Dictionary ,Tuple(Exception Handling)

Code

```
#21 to 25

def list_operations():
try:
        lst = list(map(int, input("Enter numbers separated by space: ").split()))
print("Original List:", lst)

        # Add an element
        new_element = int(input("Enter a number to add: "))
lst.append(new_element)
        print("List after addition:", lst)

        # Remove an element         remove_element =
int(input("Enter a number to remove: "))
lst.remove(remove_element)         print("List after removal:", lst)

        # Sort list
lst.sort()
        print("Sorted List:", lst)

        # Access element by index
        index = int(input("Enter index to access: "))
print(f"Element at index {index}: {lst[index]}")

    except ValueError:
        print("Invalid input! Please enter numbers only.")
except IndexError:
        print("Index out of range! Please enter a valid index.")
except Exception as e:
        print(f"An error occurred: {e}")

list_operations()




def set_operations():     try:         set1 = set(map(int, input("Enter first set
elements separated by space:
").split()))
        set2 = set(map(int, input("Enter second set elements separated by space:
").split()))

        print("Set 1:", set1)
print("Set 2:", set2)

        # Union and Intersection
```

```
        print("Union of Sets:", set1 | set2)
print("Intersection of Sets:", set1 & set2)

        # Adding and Removing elements
        new_element = int(input("Enter a number to add in Set 1: "))
set1.add(new_element)
        print("Set 1 after addition:", set1)

        remove_element = int(input("Enter a number to remove from Set 1: "))
set1.remove(remove_element)  # Raises KeyError if element not found
print("Set 1 after removal:", set1)

    except ValueError:
        print("Invalid input! Please enter numbers only.")
except KeyError:
        print("Element not found in set!")
except Exception as e:
        print(f"An error occurred: {e}")

set_operations()



def dictionary_operations():
try:
        student_dict = {}
        n = int(input("Enter number of students: "))

        for _ in range(n):
            key = input("Enter student name: ")
value = int(input(f"Enter marks for {key}: "))
student_dict[key] = value

        print("Student Dictionary:", student_dict)

        # Access value         search_key = input("Enter student
name to search: ")         print(f"Marks of {search_key}:
{student_dict[search_key]}")

        # Update value
        update_key = input("Enter student name to update marks: ")
new_marks = int(input("Enter new marks: "))
student_dict[update_key] = new_marks         print("Updated
Dictionary:", student_dict)

        # Remove key
```

```
        remove_key = input("Enter student name to remove: ")
del student_dict[remove_key]
        print("Dictionary after deletion:", student_dict)

    except ValueError:
```

```
        print("Invalid input! Please enter a valid number.")
except KeyError:
        print("Student not found in dictionary!")
except Exception as e:
        print(f"An error occurred: {e}")

dictionary_operations()




def tuple_operations():
try:
        tpl = tuple(map(int, input("Enter numbers separated by space: ").split()))
print("Tuple:", tpl)

        # Access by index
        index = int(input("Enter index to access: "))
print(f"Element at index {index}: {tpl[index]}")

        # Slicing         start, end = map(int, input("Enter start and end indices for
slicing: ").split())         print("Sliced Tuple:", tpl[start:end])

        # Finding an element
        find_element = int(input("Enter number to find: "))
        print(f"{find_element} found at index:", tpl.index(find_element))

    except ValueError:
        print("Invalid input! Please enter numbers only.")
except IndexError:
        print("Index out of range! Please enter a valid index.")
except Exception as e:
        print(f"An error occurred: {e}")

tuple_operations()
```

Output

```
Enter numbers separated by space: 5
Original List: [5]
Enter a number to add: 12
List after addition: [5, 12]
Enter a number to remove: 22
Invalid input! Please enter numbers only.
Enter first set elements separated by space:  1 2 3
Enter second set elements separated by space: 1 2 3
Set 1: {1, 2, 3}
Set 2: {1, 2, 3}
Union of Sets: {1, 2, 3}
Intersection of Sets: {1, 2, 3}
Enter a number to add in Set 1: 4
Set 1 after addition: {1, 2, 3, 4}
Enter a number to remove from Set 1: 3
Set 1 after removal: {1, 2, 4}
Enter number of students: Dushayntsinh
Invalid input! Please enter a valid number.
Enter numbers separated by space: 4444
Tuple: (4444,)
Enter index to access: 1
Index out of range! Please enter a valid index.
>>>
```