Problem 1) Simple Calculator: In Python, implement a simple calculator that does the following
.operations: summation, subtraction, multiplication, division, sqrt, power, natural log and abs
:a) Follow the instructions below
To work with the calculator, the user is asked to enter the first number, then the −
.operation, and finally, a second number if required
.Your code has to recognize the need for the second number and ask for it if required −
.After performing one operation, the calculator prints the output of the operation −
After performing one operation, the calculator must not exit. It has to start again for −
.the next operation
.The calculator will be closed if the user writes ‘e’ as any input −
Use functions to perform the operations and the appropriate conditions to prevent −
.common errors such as entering characters as one of the numbers etc
b) Run your code and provide the results for at least one example per operation

```
import math

:(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

:(def square_root(a
(return math.sqrt(a

:(def power(a, b
return a ** b

:(def natural_log(a
:if a <= 0
".return "Error! Logarithm of non-positive number is undefined
(return math.log(a

:(def absolute(a
(return abs(a
```

```
:()def calculator
:while True
("!print("Welcome to the Simple Calculator
((" :(num1 = float(input("Enter the first number (or 'e' to exit

:'if num1 == 'e
break

(" :(operation = input("Enter the operation (+, -, *, /, sqrt, pow, log, abs

:['if operation not in ['+', '-', '*', '/', 'sqrt', 'pow', 'log', 'abs
(".print("Invalid operation. Please enter a valid operation
continue

:['if operation in ['sqrt', 'log', 'abs
result = None
:'if operation == 'sqrt
(result = square_root(num1
:'elif operation == 'log
(result = natural_log(num1
:'elif operation == 'abs
(result = absolute(num1
:else
((" :num2 = float(input("Enter the second number
:'+' == if operation
(result = add(num1, num2
:'-' == elif operation
(result = subtract(num1, num2
:'*' == elif operation
(result = multiply(num1, num2
:'/' == elif operation
(result = divide(num1, num2

("print(f"Result: {result}\n

()calculator
```

https://colab.research.google.com/drive/17dHhXc9UdaL8DJ4n-jenX41uWFZZ2YyM#scrollTo=NY91WP2cPj0R

```
Welcome to the Simple Calculator!
Enter the first number (or 'e' to exit): 10
Enter the operation (+, -, *, /, sqrt, pow,
    log, abs): +
Enter the second number: 10
Result: 20.0
```

Problem 2) There are two sets of data points, where each point is represented by two features (x,
y). Assume that there are data points in each set: C1={(3, 3), (4, 3), (3, 4)}; C2={(2, 3), (2, 2), (3,
2)} and perform the followings in Python:
a) Assign the sets to an nd array using NumPy library.
b) Plot the data points. The data points in sets C1 and C2 have to be in two different colors and shapes. Label the axes and add legends as appropriate.
c) The code asks the user to new data points to C1 or C2 by entering both x and y.
d) The code updates the figures with any new points.
e) The code repeats parts (b)-(c) unless the user writes ‘e’ as the input.

```
import numpy as np
import matplotlib.pyplot as plt

# Given data points
C1 = np.array([(3, 3), (4, 3), (3, 4)])
C2 = np.array([(2, 3), (2, 2), (3, 2)])

def plot_and_update():
    plt.figure(figsize=(8, 6))

    while True:
        # Plot existing data points
        plt.scatter(C1[:, 0], C1[:, 1], color='blue', label='C1', marker='o')
        plt.scatter(C2[:, 0], C2[:, 1], color='red', label='C2', marker='s')

        plt.xlabel('X-axis')
        plt.ylabel('Y-axis')
        plt.title('Data Points')
        plt.legend()
        plt.grid(True)
        plt.show()

        choice = input("Enter the new point as 'x y' (or 'e' to exit): ")

        if choice == 'e':
            break

        try:
            new_point = np.array([list(map(float, choice.split()))])
            category = input("Add the new point to C1 or C2? (Enter 'C1' or 'C2'): ")

            if category == 'C1':
                global C1
```

```
            C1 = np.concatenate((C1, new_point))
        elif category == 'C2':
            global C2
            C2 = np.concatenate((C2, new_point))
        else:
            print("Invalid category. Please enter 'C1' or 'C2'.")

    except ValueError:
        print("Invalid input. Please enter space-separated numerical values for x and y.")

plot_and_update()
```

https://colab.research.google.com/drive/1aqbDXlRmOQmnFk5IOYtPsfYHNqUJjVbX#scrollTo=h19eKa0vRtoS

Problem 3) Dealing with image datasets - The MNIST dataset is broken up into two parts - training, and test. Each part is made up of a series of images (28 x 28 pixel images of handwritten digits) and their respective labels (values from 0 - 9, representing which digit the image corresponds to).
a) Use mnist function in keras.datasets to load MNIST dataset. Print the following: the number of images in the training and testing sets.
b) Write a function (plot_fun (images, labels)) that plots 10 images which may be 0-9 handwritten digits in a figure with 10 subplots (2 rows and 5 columns). Each subplot should have the label of the handwritten digit in the title. Note: the code has to select the images randomly with each run.
c) Plot 10 images of number 0 in 10 subplots (2 rows and 5 columns).

```
import matplotlib.pyplot as plt
from keras.datasets import mnist
import numpy as np

# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# a) Print the number of images in the training and testing sets
print(f"Number of images in the training set: {len(train_images)}")
print(f"Number of images in the testing set: {len(test_images)}")

def plot_fun(images, labels):
    plt.figure(figsize=(10, 5))
    for i in range(10):
        # Randomly select 10 images from the dataset
        idx = np.random.randint(0, len(images))
        image = images[idx]
        label = labels[idx]
```

```
        plt.subplot(2, 5, i + 1)
        plt.imshow(image, cmap='gray')
        plt.title(f"Label: {label}")
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# b) Plot 10 random images with their respective labels
plot_fun(train_images, train_labels)

# c) Plot 10 images of number 0 in 10 subplots
zero_images = train_images[train_labels == 0][:10]
zero_labels = train_labels[train_labels == 0][:10]
plot_fun(zero_images, zero_labels)
```

https://colab.research.google.com/drive/1rAuwffjDHKrkBY3TiM8E3FGa79x5RABk#scrollTo=9LMukplyTCX-