```
'''## Q1 ## Write a Python script to make a GUI-based GST Tax Finder
Calculator which takes original
            cost and net price as an input from the user and display
the GST result. Note: Calculate GST using formula:
            GST rate = ((Net Price – original cost) * 100) / original
cost,'''

# import all functions/classes from the tkinter
from tkinter import *

# Function for finding GST rate
def findGst() :

    # take a value from the respective entry boxes
    # get method returns current text as string
    org_cost= int(org_priceField.get())

    N_price = int(net_priceField.get())

    # calculate GST rate
    gst_rate = ((N_price - org_cost) * 100) / org_cost;

    # insert method inserting the
    # value in the text entry box.
    gst_rateField.insert(10, str(gst_rate) + " % ")


# Function for clearing the
# contents of all text entry boxes
def clearAll():


    # deleting the content from the entry box
    org_priceField.delete(0, END)

    net_priceField.delete(0, END)

    gst_rateField.delete(0, END)


# Driver Code
if __name__ == "__main__" :

    # Create a GUI window
    gui = Tk()

    # Set the background colour of GUI window
    gui.configure(background = "light green")
```

```
# set the name of tkinter GUI window
gui.title("GST Rate Finder")

# Set the configuration of GUI window
gui.geometry("300x300")

# Create a Original Price: label
org_price = Label(gui, text = "Original Price",
                  bg = "blue")

# Create a Net Price : label
net_price = Label(gui, text = "Net Price",
                  bg = "blue")

# Create a Find Button and attached to
# findGst function
find = Button(gui, text = "Find", fg = "Black",
              bg = "Red",
              command = findGst)

# Create a Gst Rate : label
gst_rate = Label(gui, text = "Gst Rate", bg = "blue")

# Create a Clear Button and attached to
# clearAll function
clear = Button(gui, text = "Clear", fg = "Black",
              bg = "Red",
              command = clearAll)

# grid method is used for placing
# the widgets at respective positions
# in table like structure .

# padx attributed provide x-axis margin
# from the root window to the widget.

# pady attributed provide y-axis
# margin from the widget.
org_price.grid(row = 1, column = 1,padx = 10,pady = 10)

net_price.grid(row = 2, column = 1, padx = 10, pady = 10)

find.grid(row = 3, column = 2,padx = 10,pady = 10)

gst_rate.grid(row = 4, column = 1,padx = 10, pady = 10)

clear.grid(row = 5, column = 2, padx = 10, pady = 10)

# Create a text entry box for filling or typing the information.
org_priceField = Entry(gui)
```

```
    net_priceField = Entry(gui)

    gst_rateField = Entry(gui)

    # grid method is used for placing
    # the widgets at respective positions
    # in table like structure .
    org_priceField.grid(row = 1, column = 2 ,padx = 10,pady = 10)

    net_priceField.grid(row = 2, column = 2, padx = 10,pady = 10)

    gst_rateField.grid(row = 4, column = 2, padx = 10,pady = 10)

    # Start the GUI
    gui.mainloop()

'''## Q2 ## Write a Python script to create a GUI-based Calendar
application using Tkinter module that
            display the calendar with respect to the year entered by
the user.
            Note: import Calendar library.'''

# import all methods and classes from the tkinter
from tkinter import *

# import calendar module
import calendar

# Function for showing the calendar of the given year
def showCal() :

    # Create a GUI window
    new_gui = Tk()

    # Set the background colour of GUI window
    new_gui.config(background = "white")

    # set the name of tkinter GUI window
    new_gui.title("CALENDAR")

    # Set the configuration of GUI window
    new_gui.geometry("550x600")

    # get method returns current text as string
    fetch_year = int(year_field.get())

    # calendar method of calendar module return
    # the calendar of the given year .
    cal_content = calendar.calendar(fetch_year)
```

```
    # Create a label for showing the content of the calendar
    cal_year = Label(new_gui, text = cal_content, font = "Consolas 10
bold")

    # grid method is used for placing
    # the widgets at respective positions
    # in table like structure.
    cal_year.grid(row = 5, column = 1, padx = 20)

    # start the GUI
    new_gui.mainloop()


# Driver Code
if __name__ == "__main__" :

    # Create a GUI window
    gui = Tk()

    # Set the background colour of GUI window
    gui.config(background = "white")

    # set the name of tkinter GUI window
    gui.title("CALENDAR")

    # Set the configuration of GUI window
    gui.geometry("250x140")

    # Create a CALENDAR : label with specified font and size
    cal = Label(gui, text = "CALENDAR", bg = "dark gray",
                           font = ("times", 28, 'bold'))

    # Create a Enter Year : label
    year = Label(gui, text = "Enter Year", bg = "light green")

    # Create a text entry box for filling or typing the information.
    year_field = Entry(gui)

    # Create a Show Calendar Button and attached to showCal function
    Show = Button(gui, text = "Show Calendar", fg = "Black",
                           bg = "Red", command = showCal)

    # Create a Exit Button and attached to exit function
    Exit = Button(gui, text = "Exit", fg = "Black", bg = "Red",
command = exit)

    # grid method is used for placing
    # the widgets at respective positions
```

```
    # in table like structure.
    cal.grid(row = 1, column = 1)

    year.grid(row = 2, column = 1)

    year_field.grid(row = 3, column = 1)

    Show.grid(row = 4, column = 1)

    Exit.grid(row = 6, column = 1)

    # start the GUI
    gui.mainloop()


'''## Q3 ## Write a Python script to create a GUI-based calculator
using Tkinter module, which
            can perform basic arithmetic operations addition,
subtraction, multiplication, and division.'''

from tkinter import *
root = Tk()
root.title("Calculator")
def btn_click(item):
    global expression
    expression = expression + str(item)
    input_text.set(expression)

def btn_clear():
    global expression
    expression = ""
    input_text.set("")

def btn_equal():
    global expression
    result = str(eval(expression))
    input_text.set(result)
    expression = ""

expression = ""
input_text = StringVar()

input_frame = Frame(root, width = 312, height = 50, bd = 0,
highlightbackground = "black", highlightcolor = "black",
highlightthickness = 1)
input_frame.pack(side = TOP)

input_field = Entry(input_frame, font = ('arial', 18, 'bold'),
textvariable = input_text, width = 50,fg = "black", bg = "#eee", bd =
```

```
0, justify = RIGHT)
input_field.grid(row = 0, column = 0)
input_field.pack(ipady = 10)

btns_frame = Frame(root, width = 312, height = 272.5, bg = "grey")
btns_frame.pack()

clear = Button(btns_frame, text = "Clear", fg = "black", width = 32,
height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda:
btn_clear()).grid(row = 0, column = 0, columnspan = 3, padx = 1, pady
= 1)
divide = Button(btns_frame, text = "/", fg = "black", width = 10,
height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda:
btn_click("/")).grid(row = 0, column = 3, padx = 1, pady = 1)

seven = Button(btns_frame, text = "7", fg = "black", width = 10,
height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(7)).grid(row = 1, column = 0, padx = 1, pady = 1)
eight = Button(btns_frame, text = "8", fg = "black", width = 10,
height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(8)).grid(row = 1, column = 1, padx = 1, pady = 1)
nine = Button(btns_frame, text = "9", fg = "black", width = 10, height
= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(9)).grid(row = 1, column = 2, padx = 1, pady = 1)
multiply = Button(btns_frame, text = "*", fg = "black", width = 10,
height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda:
btn_click("*")).grid(row = 1, column = 3, padx = 1, pady = 1)

four = Button(btns_frame, text = "4", fg = "black", width = 10, height
= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(4)).grid(row = 2, column = 0, padx = 1, pady = 1)
five = Button(btns_frame, text = "5", fg = "black", width = 10, height
= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(5)).grid(row = 2, column = 1, padx = 1, pady = 1)
six = Button(btns_frame, text = "6", fg = "black", width = 10, height
= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(6)).grid(row = 2, column = 2, padx = 1, pady = 1)
minus = Button(btns_frame, text = "-", fg = "black", width = 10,
height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda:
btn_click("-")).grid(row = 2, column = 3, padx = 1, pady = 1)

one = Button(btns_frame, text = "1", fg = "black", width = 10, height
= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(1)).grid(row = 3, column = 0, padx = 1, pady = 1)
two = Button(btns_frame, text = "2", fg = "black", width = 10, height
= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(2)).grid(row = 3, column = 1, padx = 1, pady = 1)
three = Button(btns_frame, text = "3", fg = "black", width = 10,
height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(3)).grid(row = 3, column = 2, padx = 1, pady = 1)
```

```
plus = Button(btns_frame, text = "+", fg = "black", width = 10, height
= 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda:
btn_click("+")).grid(row = 3, column = 3, padx = 1, pady = 1)

zero = Button(btns_frame, text = "0", fg = "black", width = 21, height
= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda:
btn_click(0)).grid(row = 4, column = 0, columnspan = 2, padx = 1, pady
= 1)
point = Button(btns_frame, text = ".", fg = "black", width = 10,
height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda:
btn_click(".")).grid(row = 4, column = 2, padx = 1, pady = 1)
equals = Button(btns_frame, text = "=", fg = "black", width = 10,
height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda:
btn_equal()).grid(row = 4, column = 3, padx = 1, pady = 1)

root.mainloop()
```

```
'''## Q4 ## Write a Python Script to create a list of marks for n
number of students entered by the user.
            Sort the input list using merge/Quick sort algorithm and
print the final sorted List.
            For e.g.: List entered by the user [23, 12, 33, 34]
            List after sorting is: [12, 23, 33, 34]'''


lst_in = eval(input("Enter the list: "))

#Using quick sort algorithm
def quick_sort(list):
    if len(list) <= 1:
#Base case
        return list
    else:
        left_arr = [x for x in list if x < list[0]]
        pivot = [x for x in list if x == list[0]]
        right_arr = [x for x in list if x > list[0]]
        return quick_sort(left_arr) + pivot + quick_sort(right_arr)
#Recursive call by taking pivot as the first element of the unsorted
list
def merge_sort(list):
    pass

lst_out = quick_sort(lst_in)
print("The sorted list(using quick sort algorithm) is: ", lst_out)
lst_out2 = merge_sort(lst_in)
print("The sorted list(using merge sort algorithm) is: ", lst_out)
```

```
Enter the list: [23,14,22,33,43,34]
The sorted list(using quick sort algorithm) is:  [14, 22, 23, 33, 34,
43]
```

```
The sorted list(using merge sort algorithm) is:  [14, 22, 23, 33, 34, 43]
```

```
'''## Q5 ## Write a Python Script to create an integer array containing duplicates entered by the user and
            then perform the following:
            a. Sort the inputted array.
            b. Using binary search algorithm Search the element entered by the user in the sorted array
                list (if the element is not present, print an error message)
            c. Count the number of occurrences of that element.
            For e.g. : Input array : [4,5,2,4,5,6,2,3,4,5,7]
            Sorted array :[2,2,3,4,4,4,5,5,5,6,7]
            Number of occurrences of element 4 is: 3'''

'''Sorting, binary search and counting occurrences of element.'''

print("\n\tBINARY SEARCH\n")


def binary_search(array, x):
    """Defining a function to search an element
        in an array using binary search algorithm.
        array - array from which element will be found.
        x - element which will be searched in the array"""

    lower = 0
    higher = len(array) - 1

    # using while loop to change the position of pointer until they meet each other.
    while lower <= higher:

        mid = (lower + higher) // 2
# Finding middle element in the portion array.

        if array[mid] == x:
# If middle element = required element returning it.
            return mid

        elif array[mid] < x:
# if middle element < required element, changing lower value to mid+1.
            lower = mid + 1

        elif array[mid] > x:
# if middle element > required element, changing higher value to mid-1
            higher = mid - 1
```

```
    return "not found"
# if element in not found in the array returning "not found"


while True:
# using while loop and exceptions to ensure correct input.
    try:
        arr = list(map(int, input("Enter the array (Enter values
separated by space) : ").split()))                    # taking array
input from user.
        break
    except Exception as e:
        print(f"\nError! : {e}")
# if any error occurs, printing it and taking input again.
        continue

print()
print("Array entered by the user :", arr)

# part a.
'''sorting the array.'''

print("\n\tPart a : sorting the array\n")

arr.sort()
# Sorting the array.
print("Sorted Array :", arr)

# part b.
'''Using binary search to find an element in the array.'''

print("\n\tPart b : binary search\n")

while True:
# using while loop and exceptions to ensure correct input.
    try:
        fnd = int(input("Enter the number you want to find in the
array : "))                                          # asking user
for the element which needs to be searched.
        break
    except Exception as e:
        print(f"\nError! : {e}")
# if any error occurs, printing it and taking input again.
        continue

element = binary_search(arr, fnd)
# element = index of the element in the array.

print()
```

```python
if element == "not found":
    print(f"Element {fnd} was NOT FOUND in given array.")
# if element is not found in the array printing not found.
else:
    print(f"Element {fnd} was found at the index : {element}.")

# part c.
'''counting the occurrence of the element found above.'''
print("\n\tPart c : counting occurrences\n")

num = arr.count(fnd)
# finding number of occurrence using .count

print(f"Number of occurrences of element {fnd} is : {num}")

print("-" * 80)
```

```
     BINARY SEARCH

Enter the array (Enter values separated by space) : 69 1 4 3 69 69 456
69 45356 69

Array entered by the user : [69, 1, 4, 3, 69, 69, 456, 69, 45356, 69]

     Part a : sorting the array

Sorted Array : [1, 3, 4, 69, 69, 69, 69, 69, 456, 45356]

     Part b : binary search

Enter the number you want to find in the array : 69

Element 69 was found at the index : 4.

     Part c : counting occurrences

Number of occurrences of element 69 is : 5
--------------------------------------------------------------------------
----------
```

```python
'''## Q6 ## Write a Python Script to remove duplicate numbers from the
list of integer numbers and also
            write functions to sort the list (after removal of
duplicate numbers) using selection and bubble sort.
            For e.g. : Input array : [4,5,2,4,5,6,5,4,5,5,6]
            Sorted array :[2,4,5,6]'''
```

```python
# Python code to remove duplicate elements
def Remove(duplicate):
    final_list = []
    for num in duplicate:
        if num not in final_list:
            final_list.append(num)
    return final_list

# Driver Code
duplicate = [2, 4, 10, 20, 5, 2, 20, 4]
print(Remove(duplicate))
```

```
[2, 4, 10, 20, 5]
```