Images for this project, if you can’t find your own, clips these as SwitchOff.png and SwitchOn.png

```
#Written by Terry Smith July 2023
from tkinter import *
from tkinter import messagebox #needed to avoid Warning that "messagebox is not defined"
import array as arr
from typing import List

my_win = Tk() #create a TKinter window object
my_win.geometry("370x350") # set width and height of the window
my_win.title('ASCII / Binary convertor') # set window caption
#my_win.config(bg="skyblue")  # specify background color

# Create frames
header_frame = Frame(my_win, width=370, height=50)
header_frame.grid(row=0, column=0, padx=10, pady=5)

#main_frame = Frame(my_win, width=370, height=200, bg='grey')
main_frame = Frame(my_win, width=370, height=200)
main_frame.grid(row=1, column=0, padx=10, pady=5)

ASCII2Bin_frame = Frame(my_win, width=370, height=50)
ASCII2Bin_frame.grid(row=2, column=0, padx=0, pady=0)

footer_frame = Frame(my_win, width=370, height=50)
footer_frame.grid(row=3, column=0, padx=0, pady=0)

label_grid_rows = 7 # includes the top "title" row
label_grid_columns = 9 # includes the left-most "title" column
my_labels: List[Label] = [] # list of labels to allow updating
my_buttons: List[Button] = [] # list of buttons

BinaryArray = arr.array('i', [0] * 8) #i specifies an array of 8 unsigned integers, each set to 0

def get_label_index(row, col):
    #function to return the index in the my_labels list for a label specified by its row and column
positions
    #print(row * label_grid_columns + col)
    return row * label_grid_columns + col

def display_exponents():
    #Display the 8 exponents of 2 from 0 to 7 in labels in row 0
    exponent_count = 8
    my_row = 0
    for p in range(exponent_count):
        my_labels[get_label_index(my_row, p + 1)].configure(text = exponent_count - (p + 1))

def display_powers_of_2():
    #Display 2 raised to the power from 0 to 7 in labels in the second row
    exponent_count = 8
    my_row = 1
    for p in range(exponent_count):
        my_labels[get_label_index(my_row, p + 1)].configure(text = 2 ** (exponent_count - (p + 1)))

def create_grid():
    #NOTE: This creates unused labels that are covered by the buttons
    for my_row in range(label_grid_rows):
        for my_column in range(label_grid_columns):
            #my_label = tk.Label(my_win, text = str(my_row) + ', ' +str(my_column))
            my_label = Label(main_frame, text = '')
            my_label.grid(row = my_row, column = my_column, padx = 5, pady = 5)
            #left align first column
            if my_column == 0:
                my_label.grid(row = my_row, column = 0, sticky = 'w')
            my_labels.append(my_label) # add the reference

def create_buttons():
```

```
    button_row = 2
    exponent_count = 8
    for button_col in range(0, exponent_count):
        #use lambda function to pass the parameter to the function "toggle"
        #append "# type: ignore" to the end of the line to suppress type checking on late binding
        my_button = Button(main_frame, image = off, bd = 0, command = lambda col = button_col:
toggle(col)) # type: ignore
        my_button.grid(row = button_row, column = button_col + 1)
        my_button.lift() # make the button visible, rather than the label created below it.
        my_buttons.append(my_button) # add the button to the list of buttons

# Define a function to respond to a button press
def toggle(button_no):
    #function to toggle BinaryArray element corresponding to the button_no and update the button
image
    BinaryArray[button_no] = 1 - BinaryArray[button_no]
    if BinaryArray[button_no] == 0:
        my_buttons[button_no].config(image = off)
    else:
        my_buttons[button_no].config(image = on)
    show_results()

def set_title():
    Label(header_frame, text = 'Binary / ASCII convertor', font='Helvetica 18 bold').grid(row=0,
column=0)

def set_row_titles():
    my_labels[get_label_index(0,0)].configure(text = 'Power of 2')
    my_labels[get_label_index(1,0)].configure(text = '2^power')
    my_labels[get_label_index(2,0)].configure(text = 'Switches')
    my_labels[get_label_index(3,0)].configure(text = 'Binary')
    my_labels[get_label_index(4,0)].configure(text = 'Values')
    my_labels[get_label_index(5,0)].configure(text = 'Sum of values')
    my_labels[get_label_index(6,0)].configure(text = 'ASCII Character')

def show_results():
    #Create labels and assign to grid for future updates
    exponent_count = 8
    binary_row = 3
    power_of_two_row = 4
    my_column = 1
    sum_of_powers = 0
    for p in range(exponent_count):
        #display binary values
        my_labels[get_label_index(binary_row, p + 1)].configure(text = BinaryArray[p])
        #calculate the power of two corresponding to the value of the BinaryArray element
        value = BinaryArray[p] * (2 ** (exponent_count - 1 - p))
        my_labels[get_label_index(power_of_two_row, p + 1)].configure(text = value)
        #sum the values of the powers of 2
        sum_of_powers = sum_of_powers + value
        #increment column counter
        my_column = my_column + 1 # move to next column
    #Now the loop has finished, display the sum of the powers of two
    my_labels[get_label_index(power_of_two_row + 1, 1)].configure(text = sum_of_powers)
    #and the corresponding ASCII character
    my_labels[get_label_index(power_of_two_row + 2, 1)].configure(text = chr(sum_of_powers))

######################################################################

def ASCII2Bin():
    global BinaryArray
    Chr = ASCIIChar.get()
    if Chr.strip() > '':
#    BinaryArray = BinStr2Array(ASCII2BinStr(ASCIIChar.get())) # ASCIIChar is a StringVar defined
below
        BinaryArray = BinStr2Array(ASCII2BinStr(Chr)) # ASCIIChar is a StringVar defined below
        show_results() #Refresh display
        UpdateButtonDisplay() #Update Button display to reflect entered character
    else:
        messagebox.showwarning(title="Warning", message="Please enter an ASCII character")

def divideBy2(decNumber):
    Str = '' #Initialise the string that will hold the binary number
    while decNumber > 0:
        remainder = decNumber % 2 #get the remainder after dividing by 2 (0 or 1)
        Str = str(remainder) + Str #Add the remainder value to the beginning of the string
        decNumber = decNumber // 2 # get the integer part of decNumber / 2
    return Str #return the decimal string. NB: Length of Str depends on value of decNumber

def ASCII2BinStr(ASCIIChr):
```

```
    return ('00000000' + divideBy2(ord(ASCIIChr)))[-8:]

def BinStr2Array(BinStr):
    for n in range(8):
        #print(BinStr[n])
        if BinStr[n] == '0':
            BinaryArray[n] = 0
        else:
            BinaryArray[n] = 1
    return BinaryArray

# Define a function to update button positions
def UpdateButtonDisplay():
    #function to set button position according to value of BinaryArray element
    for button_no in range(8):
        if BinaryArray[button_no] == 0:
            my_buttons[button_no].config(image = off)
        else:
            my_buttons[button_no].config(image = on)
    show_results()

# Create button to close the window.
def create_quit_button():
    Button(footer_frame, text="Quit", command = my_win.destroy).grid(row = 0, column = 0) # use
destroy rather than quit to actually close the window

######################################################################
# Define Images for buttons
on = PhotoImage(file = "SwitchOn.png")
off = PhotoImage(file = "SwitchOff.png")
# Create the grid to hold all widgets
create_grid()
# Create toggle buttons to go in row 3 of the grid
create_buttons()
#Now we can initialise display values
set_title()
set_row_titles()
display_exponents()
display_powers_of_2()

Label(ASCII2Bin_frame, text = 'ASCII character to binary:').grid(row = 0, column = 0)
ASCIIChar = StringVar() #Definer StringVar to get text from  EditBox
Entry(ASCII2Bin_frame, textvariable = ASCIIChar).grid(row = 0, column = 1)
Button(ASCII2Bin_frame, text="Convert", command = ASCII2Bin).grid(row = 0, column = 2)
create_quit_button()

#Main loop to run the program
my_win.mainloop()
```