POP-project

Professional skills 1

Jonas Baelus

Academiejaar 2021-2022

Campus : Thomas More Geel

INHOUDSOPGAVE

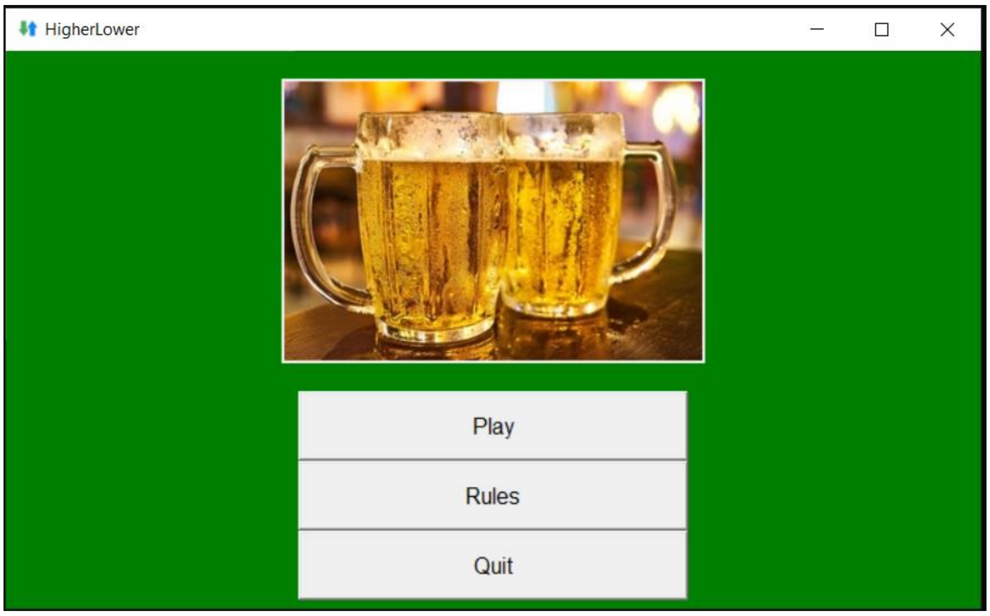
Code

```
from tkinter import *
import random
from PIL import Image, ImageTk


def delete_1():
    screen3.destroy()


def delete_2():
    screen4.destroy()


def delete_3():
    screen6.destroy()


def delete_4():
    screen7.destroy()


def delete_5():
    screen8.destroy()


def waslower():
    global screen3
    screen3 = Toplevel(root)
    screen3.title(f'Wrong! It was lower!')
    screen3.iconbitmap(f'images/updown.ico')
    screen3.geometry("500x200")
    Label(screen3, text=players[0].get(), font=("Helvetica", 12)).pack()
    Label(screen3, text="Take 2 sips of your drink!", font=("Helvetica",
12)).pack()
    Button(screen3, text="DONE", font=("Helvetica", 12), command =
delete_1).pack(pady=20)


def washigher():
    global screen4
    screen4 = Toplevel(root)
    screen4.title(f'Wrong! It was higher!')
    screen4.iconbitmap(f'images/updown.ico')
    screen4.geometry("500x200")
    Label(screen4, text=players[0].get(), font=("Helvetica", 12)).pack()
    Label(screen4, text="Take 2 sips of your drink!", font=("Helvetica",
12)).pack()
    Button(screen4, text="DONE", font=("Helvetica", 12), command =
delete_2).pack(pady=20)


def wasnotsame():
    global screen6
    screen6 = Toplevel(root)
    screen6.title(f'Wrong! It was not the same!')
    screen6.iconbitmap(f'images/updown.ico')
    screen6.geometry("500x200")
    Label(screen6, text=players[0].get(), font=("Helvetica", 12)).pack()
    Label(screen6, text="Take 3 sips of your drink!", font=("Helvetica",
12)).pack()
    Button(screen6, text="DONE", font=("Helvetica", 12),
command=delete_3).pack(pady=20)


def wasthesame():
    global screen7
    screen7 = Toplevel(root)
    screen7.title(f'Congrats! It was the same!')
    screen7.iconbitmap(f'images/updown.ico')
    screen7.geometry("500x200")
    Label(screen7, text=players[0].get(), font=("Helvetica", 12)).pack()
    Label(screen7, text="Choose a player that needs to do AD FUNDUM!",
font=("Helvetica", 12)).pack()
    Button(screen7, text="DONE", font=("Helvetica", 12),
command=delete_4).pack(pady=20)


def rules():
    global screen8
    screen8 = Toplevel(root)
    screen8.title(f'Rules')
    screen8.iconbitmap(f'images/updown.ico')
    screen8.geometry("1000x400")
    Label(screen8, text="Rules:", font=("Helvetica", 15)).pack()
    Label(screen8, text="You have to guess whether the new card (facing
down) is higer, lower or the same than the card shown.",
font=("Helvetica", 12)).pack()
    Label(screen8, text="When you guess higher or lower and you are wrong,
you need to take 2 sips of your drink.", font=("Helvetica", 12)).pack()
    Label(screen8, text="When you guess the same and you are wrong, you
need to take 3 sips of your drink.", font=("Helvetica", 12)).pack()
    Label(screen8, text="Guessing correctly equals nothing, unless you
guessed the same,", font=("Helvetica", 12)).pack()
    Label(screen8, text="after which you may give an AD FUNDUM to another
player.", font=("Helvetica", 12)).pack()
    Label(screen8, text="If the deck is empty, click the button 'Shuffle
Deck'.", font=("Helvetica", 12)).pack(pady=10)
    Label(screen8, text="2 = lowest         ace = highest",
font=("Helvetica", 12)).pack()
    Button(screen8, text="UNDERSTOOD", command=delete_5).pack(pady=20)


def register():
    screen1 = Toplevel(root)
    screen1.title("Register")
    screen1.iconbitmap(f'images/updown.ico')
    screen1.geometry("1200x800")
    global username
    global username_entry
    global players
    username=StringVar()
    number = int(number_players.get())
    players=[]
    for i in range(number):
        Label(screen1, text="Username:", font=("Helvetica", 12)).pack()
        Label(screen1, text="").pack()
        username_entry = Entry(screen1)
        username_entry.pack()
        players.append(username_entry)
    Button(screen1, text="Register", font=("Helvetica", 12), width =10,
height=1, command = game).pack(pady=20)


def game():
    global screen2
    screen2 = Toplevel(root)
    screen2.title('HigherLower - Card Deck')
    screen2.iconbitmap(f'images/updown.ico')
    screen2.geometry("1200x800")
    screen2.configure(background="green")
    # Resize Cards

    def resize_cards(card):
        # Open the image
        our_card_img = Image.open(card)

        # Resize The Image
        our_card_resize_image = our_card_img.resize((150, 218))

        # output the card
        global our_card_image
        our_card_image = ImageTk.PhotoImage(our_card_resize_image)

        # Return that card
        return our_card_image

    # Shuffle The Cards
    def shuffle():
        # Define Our Deck
        suits = ["diamonds", "clubs", "hearts", "spades"]
        values = range(2, 15)
        # 11 = Jack, 12=Queen, 13=King, 14 = Ace

        global deck
        deck = []

        for suit in suits:
            for value in values:
                deck.append(f'{value}_of_{suit}')

        # Create our players
        global dealer, player, counter
        dealer = []
        player = []
        counter = 0

        # Grab a random Card For Dealer
        global card_left
        card_left = random.choice(deck)
        # Remove Card From Deck
        deck.remove(card_left)
        # Append Card To Dealer List
        dealer.append(card_left)
        # Output Card To Screen
        global dealer_image
        dealer_image = resize_cards(f'images/cards/{card_left}.png')
        dealer_label.config(image=dealer_image)

        # Output Card To Screen
        global player_image
        player_image = resize_cards(f'images/kaart.png')
        player_label.config(image=player_image)

        # Put number of remaining cards in title bar
        screen2.title(f'HigherLower - {len(deck)} Cards Left')

        # current player
        global current_player
        current_player = players[0].get()

    # Deal Out Cards
    def higher():
        try:
            global counter
            # Grab a random Card For Dealer
            new_card = random.choice(deck)
            # Remove Card From Deck
            deck.remove(new_card)
            # Append Card To Dealer List
            dealer.append(new_card)
            kaart = dealer[counter]
            value_newcard = int(new_card.split("_", 1)[0])
            value_card = int(kaart.split("_", 1)[0])
            counter += 1

            # Is the card higher?
            if value_newcard > value_card:
                # Output Card To Screen
                global dealer_image
                dealer_image =
resize_cards(f'images/cards/{new_card}.png')
                dealer_label.config(image=dealer_image)

                # Output Card To Screen
                global player_image
                player_image = resize_cards(f'images/kaart.png')
                player_label.config(image=player_image)

                # Put number of remaining cards in title bar
                screen2.title(f'HigherLower - {len(deck)} Cards Left')

                # change player
                players.append(players[0])
                del players[0]
            else:
                dealer_image =
resize_cards(f'images/cards/{new_card}.png')
                dealer_label.config(image=dealer_image)
                screen2.title(f'HigherLower - {len(deck)} Cards Left')
                # Load page that you were wrong
                waslower()

                # change player
                players.append(players[0])
                del players[0]


        except:
            screen2.title(f'HigherLower - No Cards In Deck')

    def same():
        try:
            global counter
            # Grab a random Card For Dealer
            new_card = random.choice(deck)
            # Remove Card From Deck
            deck.remove(new_card)
            # Append Card To Dealer List
            dealer.append(new_card)
            kaart = dealer[counter]
            value_newcard = int(new_card.split("_", 1)[0])
            value_card = int(kaart.split("_", 1)[0])
            counter += 1

            # Is the card the same?
            if value_newcard == value_card:
                # Output Card To Screen
                global dealer_image
                dealer_image =
resize_cards(f'images/cards/{new_card}.png')
                dealer_label.config(image=dealer_image)

                # Output Card To Screen
                global player_image
                player_image = resize_cards(f'images/kaart.png')
                player_label.config(image=player_image)

                # Put number of remaining cards in title bar
                screen2.title(f'HigherLower - {len(deck)} Cards Left')

                #load page that you were right
                wasthesame()

                # change player
                players.append(players[0])
                del players[0]

            else:
                dealer_image =
resize_cards(f'images/cards/{new_card}.png')
                dealer_label.config(image=dealer_image)
                screen2.title(f'HigherLower - {len(deck)} Cards Left')
                #Load page that you were wrong
                wasnotsame()

                # change player
                players.append(players[0])
                del players[0]

        except:
            screen2.title(f'HigherLower - No Cards In Deck')

    def lower():
        try:
            global counter
            # Grab a random Card For Dealer
            new_card = random.choice(deck)
            # Remove Card From Deck
            deck.remove(new_card)
            # Append Card To Dealer List
            dealer.append(new_card)
            kaart = dealer[counter]
            value_newcard = int(new_card.split("_", 1)[0])
            value_card = int(kaart.split("_", 1)[0])
            counter += 1

            # Is the new card lower?
            if value_newcard < value_card:
                # Output Card To Screen
                global dealer_image
                dealer_image =
resize_cards(f'images/cards/{new_card}.png')
                dealer_label.config(image=dealer_image)

                # Output Card To Screen
                global player_image
                player_image = resize_cards(f'images/kaart.png')
                player_label.config(image=player_image)

                # Put number of remaining cards in title bar
                screen2.title(f'HigherLower - {len(deck)} Cards Left')

                #change player
                players.append(players[0])
                del players[0]

            else:
                dealer_image =
resize_cards(f'images/cards/{new_card}.png')
                dealer_label.config(image=dealer_image)
                screen2.title(f'HigherLower - {len(deck)} Cards Left')
                #Load page that you were wrong
                washigher()

                # change player
                players.append(players[0])
                del players[0]

        except:
            screen2.title(f'HigherLower - No Cards In Deck')

    my_frame = Frame(screen2, bg="green")
    my_frame.pack(pady=20)

    # Create Frames For Cards
    dealer_frame = Frame(my_frame, bd=0)
    dealer_frame.pack(side=LEFT)

    player_frame = Frame(my_frame, bd=0)
    player_frame.pack(side=RIGHT)

    higher_button = Button(my_frame, text="Higher!", font=("Helvetica",
14), command=higher)
    higher_button.pack(side=LEFT, padx=20)

    same_button = Button(my_frame, text="Equal!", font=("Helvetica", 14),
command=same)
    same_button.pack(side=LEFT, padx=20)

    lower_button = Button(my_frame, text="Lower!", font=("Helvetica", 14),
command=lower)
    lower_button.pack(side=LEFT, padx=20)

    # Put cards in frames
    dealer_label = Label(dealer_frame, text='')
    dealer_label.pack(pady=20)

    player_label = Label(player_frame, text='')
    player_label.pack(pady=20)

    # Buttons
    shuffle_button = Button(screen2, text="Shuffle Deck",
font=("Helvetica", 14), command=shuffle)
    shuffle_button.pack(pady=20)

    Button(screen2, text="Stop playing", font=("Helvetica", 14),
command=root.destroy).pack(pady=20)

    # Shuffle Deck On Start

    shuffle()


def aantal():
    screen5 = Toplevel(root)
    screen5.title("Amount of players")
    screen5.iconbitmap(f'images/updown.ico')
    screen5.geometry("1200x300")

    global number_players
    number_players = StringVar()

    Label(screen5, text="With how many players do you want to play?",
font=("Helvetica", 12)).pack()

    username_entry = Entry(screen5, textvariable=number_players)
    username_entry.pack()

    Button(screen5, text="Next", font=("Helvetica", 12), height="2",
width="30", command=register).pack()


def main_screen():
    global root
    root = Tk()
    root.title('HigherLower')
    root.iconbitmap(f'images/updown.ico')
    # root.geometry("900x500")
    root.geometry("700x400")
    root.configure(background="green")

    frame_image = Frame(root, bg="green", relief=SUNKEN)
    frame_image.pack(side=TOP, fill="x")

    frame_image.picture = PhotoImage(file=f'images/beer.png')
    frame_image.label = Label(frame_image, image=frame_image.picture)
    frame_image.label.pack(pady=20)


    Button(text = "Play", font=("Helvetica", 12), height = "2", width
="30", command = aantal).pack()
    Button(text="Rules", font=("Helvetica", 12), height="2", width="30",
command=rules).pack()
    Button(text="Quit", font=("Helvetica", 12), height="2", width="30",
command=root.destroy).pack()
```

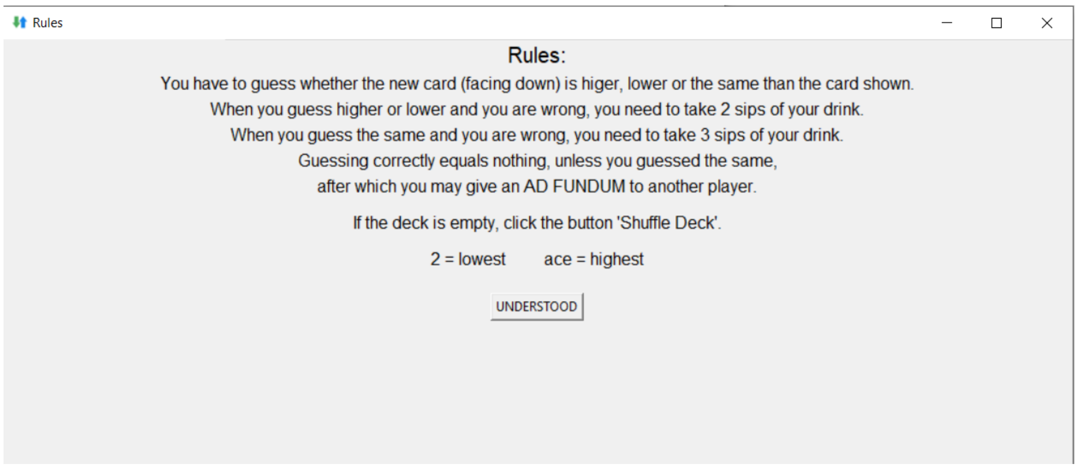
```
    root.mainloop()

main_screen()
```

UITLEG CODE

1) Het main programma bestaat maar uit 1 functie die wordt opgeroepen, nl. mainscreen()
2) Deze functie maakt de eerste pagina aan die 3 knoppen bevat (play, rules , quit)



3) De quit button stopt het programma
4) De rules button roept de rules() functie op die een nieuwe pagina opent waar de regels weergegeven worden en een button ("understood") die de pagina weer sluit

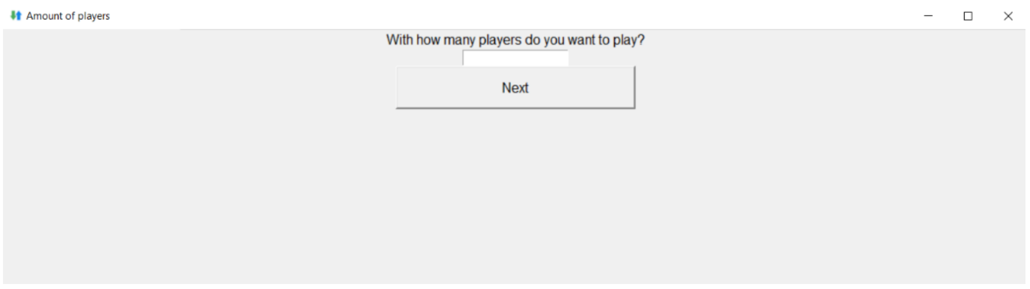


Rules:
You have to guess whether the new card (facing down) is higer, lower or the same than the card shown.
When you guess higher or lower and you are wrong, you need to take 2 sips of your drink.
When you guess the same and you are wrong, you need to take 3 sips of your drink.
Guessing correctly equals nothing, unless you guessed the same,
after which you may give an AD FUNDUM to another player.

If the deck is empty, click the button 'Shuffle Deck'.

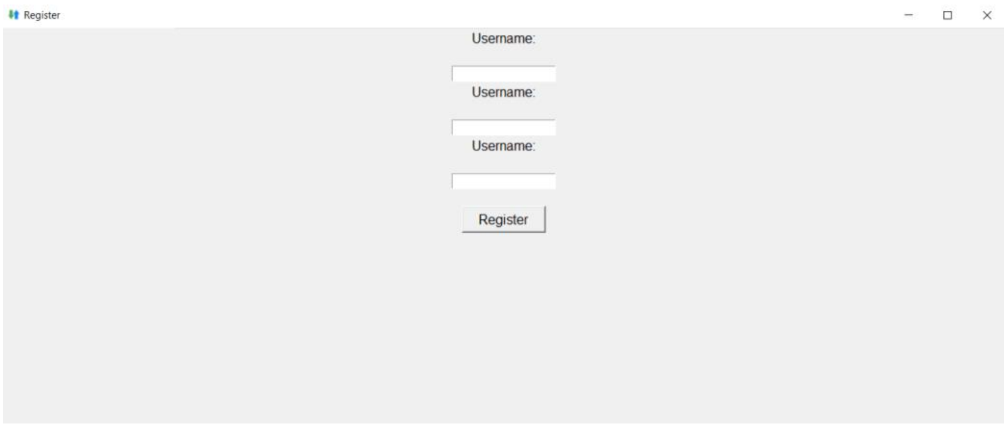
2 = lowest   ace = highest

5) De play button start de functie aantal() die een nieuwe pagina opent

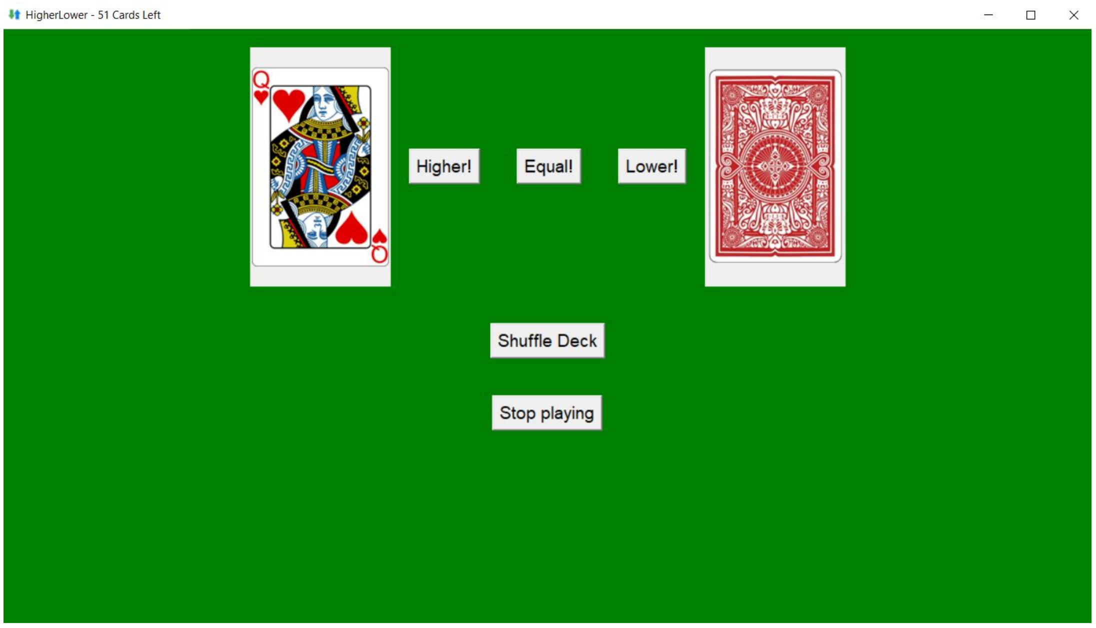


With how many players do you want to play?

6) Op de pagina worden het aantal spelers gevraagd
7) Op de pagina is ook een next button die de functie register() start
8) De functie register maakt een pagina aan waar de usernames gevraagd worden



9) De register button start de functie game()
10) De functie game maakt een nieuwe pagina waar het effectieve spel gespeeld wordt.



11) De functie game bevat enkele functies (resize_cards(), shuffle(), higher(), same() en lower())
12) De pagina bevat enkele buttons die de functies oproepen:
- Higher! Button: higher()
- Equal! Button: same()
- Lower! Button: lower()
- Shuffle Deck Button: shuffle()
- Stop playing Button: doet het spel dicht

LINK DEMONSTRATIE

https://youtu.be/afMAkA2Ckyk

BRONNEN

1) *Create UI in Python-Tkinter*, van tutorialsteacher.com: https://www.tutorialsteacher.com/python/create-gui-using-tkinter-python

2) *Python - Tkinter pack() Method*, van tutorialspoint.com: https://www.tutorialspoint.com/python/tk_pack.htm

3) Codemy.com. (2022, 18 januari). *Create A Deck Of Cards And Deal Them Out - Python Tkinter GUI Tutorial 206* [Video]. YouTube. https://www.youtube.com/watch?v=xJZksz2UpqE&t=1307s

4) johan godinho. (2018a, september 25). *How to create a graphical register and login system in python using Tkinter* [Video]. YouTube. https://www.youtube.com/watch?v=Xt6SqWuMSA8

5) johan godinho. (2018, 27 september). *How to create a graphical register and login system in python using Tkinter Part 2* [Video]. YouTube. https://www.youtube.com/watch?v=Z-deSpgtIG0