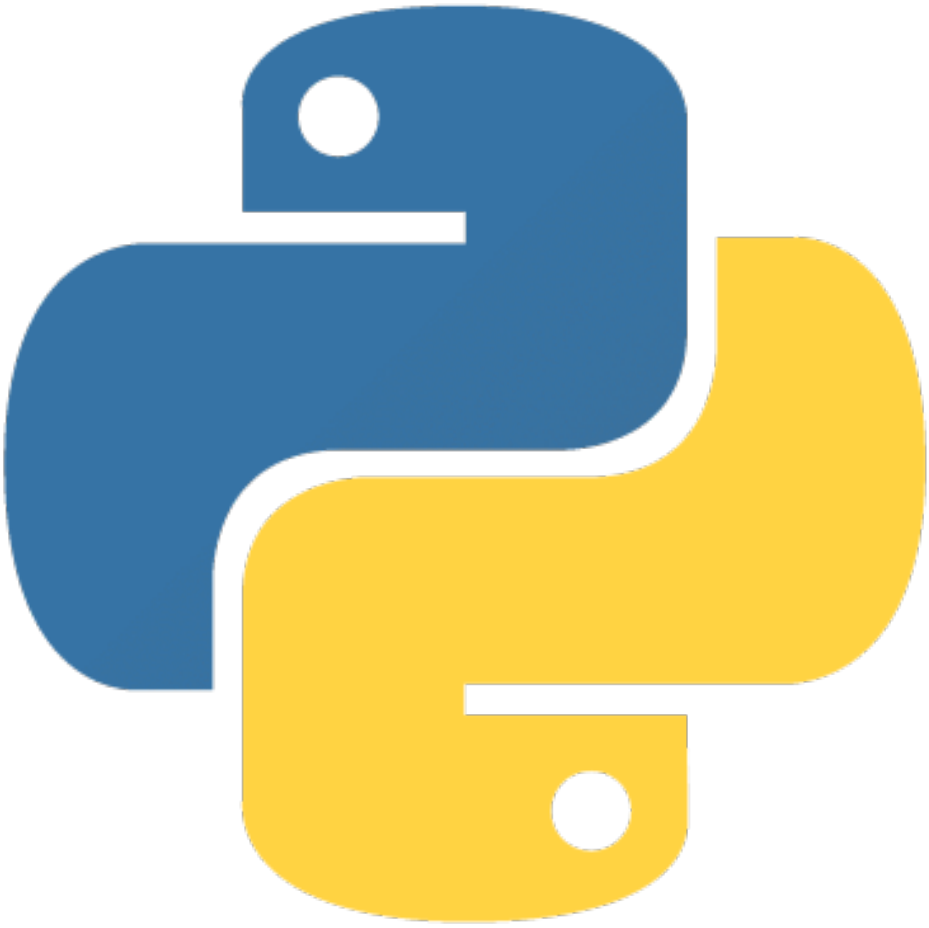


python™

Eindproject Python E – Pluk van den Donker

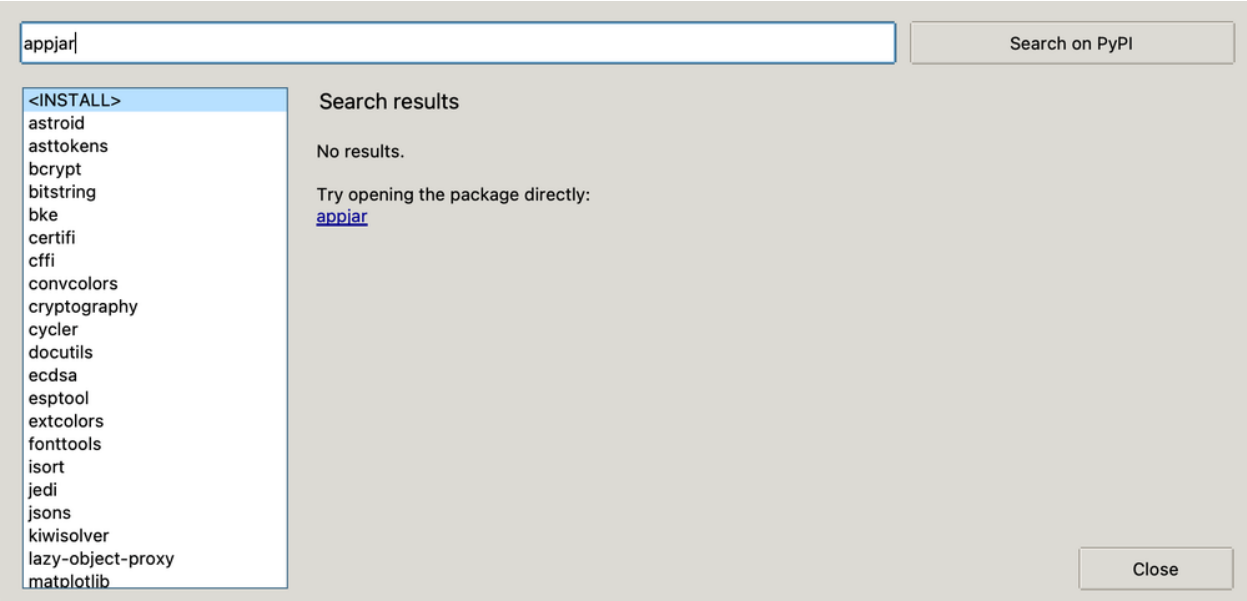
Table of Contents

Hoe runt u het programma

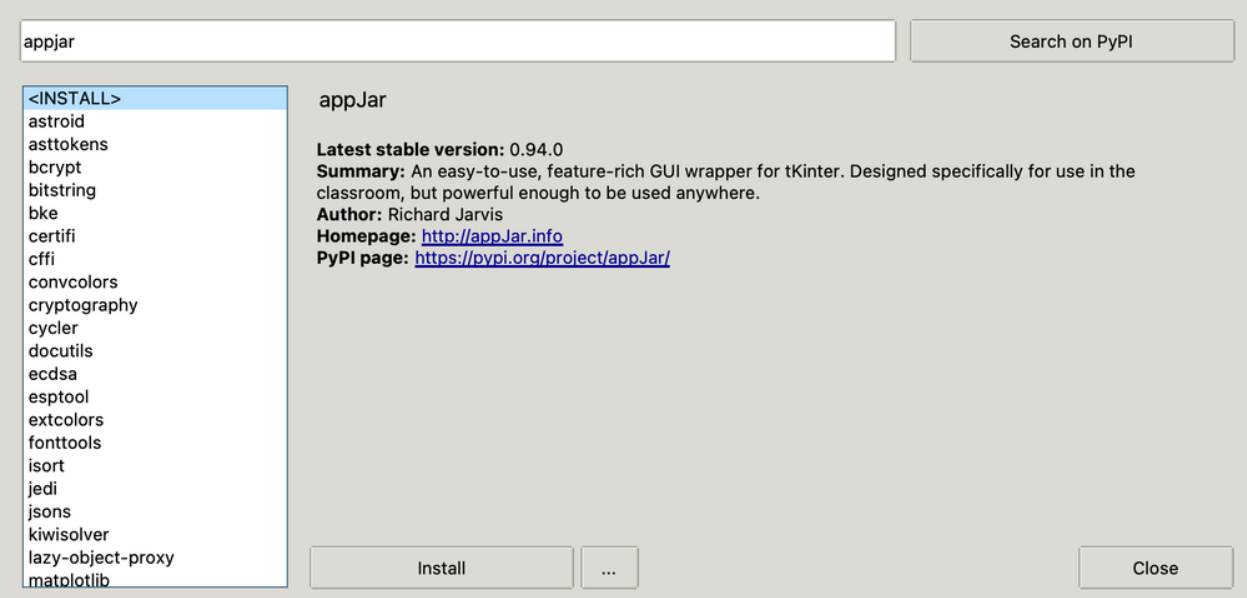
Thonny

Installeer Thonny hier: https://thonny.org/

In Thonny kun je een externe bibliotheek toevoegen door bovenin op "Tools" en dan op "Manage packages..." te klikken. Type in het zoekveld de naam appjar in en klik op "Search on PyPI". Er verschijnt een melding, dat er geen resultaten zijn.

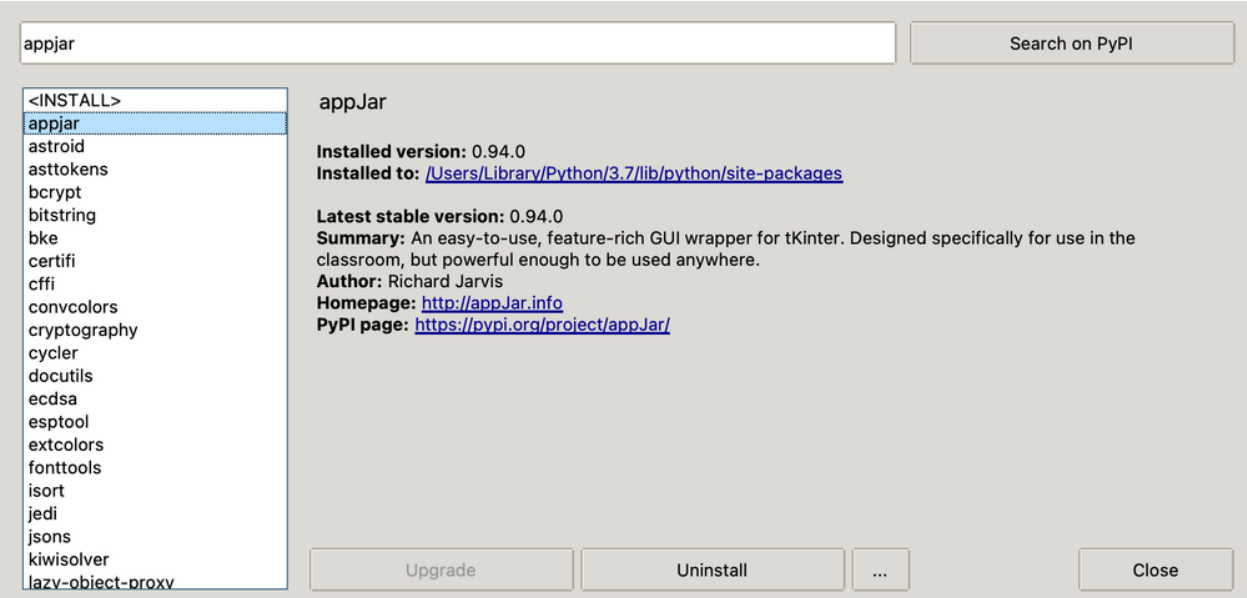


Klik onder de tekst "Try opening the package directly" op de tekst "appjar". Nu wordt er wel een resultaat gevonden.



Klik op de knop "Install", de bibliotheek wordt nu geïnstalleerd. Dit kan even duren! Wacht rustig af.

Nadat de bibliotheek is geïnstalleerd, verschijnt er geen melding. Je kunt echter wel zien dat het gelukt is, omdat appjar nu in de lijst aan de linkerkant staat..



Nu dat je Thonny hebt geinstalleerd, kan je main.py en main_no_simulation openen met Thonny. In de menubalk staat een knop met een groen pijltje genaamd ‘run’ en als u daarop drukt wordt de code uitgevoerd

Als u python al heeft geïnstaleerd

1. Open en run ‘installAppJar.bat’
2. Open en run ‘start.bat’ om de python file te starten

   Als dit niet werkt installeer Thonny

Code uitgelegd

Code

```
from appJar import gui
import random

class Klant():
    def __init__(self, kinderen, volwassenen, studenten65, films):
        self.kinderen = int(kinderen)
        self.volwassenen = int(volwassenen)
        self.studenten65 = int(studenten65)

        self.film = random.choice(list(films.keys()))
        self.tijdslot = random.choice(list(films[self.film].keys()))
```

maakt class Klant wat een klant voorstelt voor de simulatie en analyse. De film wordt willekeurig gekozen uit films, en het tijdslot willekeurig bij de film

```
def simuleer(aantalKlanten, films):
    klanten = []
    for i in range(aantalKlanten):
        klanten.append(Klant(random.randint(0, 3),random.randint(0,
3),random.randint(0, 3),films))
    return klanten
```

maakt aantalKlanten klanten met een willekeurig aantal kinderen, volwassenen en studenten/65+’ers tussen de 0 en 3, en een willekeurige film en tijdslot

```
app=gui('Bioscoopticketautomaat','600x300')

simulerend=True # of er gesimuleerd moet worden
filmCapaciteiten = {
    'Oppenheimer': {
        '13:00': 20,
        '15:00': 25,
        '17:00': 25,
    },
    'Dune': {
        '18:30': 25,
        '19:00': 25,
        '20:00': 30,
    },
    'Avatar': {
        '8:00': 20,
        '12:00': 15,
        '16:00': 30,
```

```
    }
}
uitverkocht={
    'Oppenheimer': 0,
    'Dune': 0,
    'Avatar': 0
}
```

Filmcapaciteiten en uitverkochte films

```
def koop(name,gesimuleerd=False,klant=None):
    global filmCapaciteiten,uitverkocht

    if gesimuleerd and klant:
        filmkeuze=klant.film
        tijdslot=klant.tijdslot
        kind=klant.kinderen
        volwassene=klant.volwassenen
        student65=klant.studenten65
    else:

        filmkeuze=str(app.getOptionBox('filmkeuzes'))
        tijdslot=str(app.getOptionBox('tijdslot'))
        kind=int(app.getSpinBox('kind'))
        volwassene=int(app.getSpinBox('volwassene'))
        student65=int(app.getSpinBox('student/65+'))
        klanten.append(Klant(kind,volwassene,student65,filmCapaciteiten))
```

kijkt of de klant gesimuleerd is of echt, zodat het de waardes uit de gui of de Klant class moet halen

```
    if filmkeuze:
        capaciteit=filmCapaciteiten[filmkeuze][tijdslot]
    else:
        capaciteit=0
```

pakt de capaciteit van de geselecteerde film

```
    if kind+volwassene+student65<=capaciteit:
        filmCapaciteiten[filmkeuze][tijdslot]-=kind+volwassene+student65
        if not gesimuleerd:
            app.setLabel('capaciteit',f'Nog {capaciteit} plaatsen beschikbaar')
            app.setLabel('intro','Transactie succesvol!')
            app.after(3000, lambda: app.setLabel('intro','Bioscoopticketautomaat'))
```

als er genoeg plaatsen zijn voor de mensen laat dit de transactie door en meldt dat de transactie successvol is voor 3 seconden

```
        with open('Eindopdracht E/tickets.txt','a') as f:
            f.write(f'{filmkeuze}: {kind} kinderen, {volwassene} volwassenen,
{student65} studenten/65+\'ers totale prijs: €{kind*5+volwassene*8+student65*6}.
Plaatsen over: {capaciteit}\n')
```

het zet de transactie in het logboek

```
    else:
        uitverkocht[filmkeuze]+=1
        if not gesimuleerd:

            app.setLabel('intro','Transactie onsuccesvol! Er zijn niet genoeg plaatsen
bij de film.')
            app.after(5000, lambda: app.setLabel('intro','Bioscoopticketautomaat'))
        with open('Eindopdracht E/tickets.txt','a') as f:
            f.write(f'GEFAALD, TE WEINIG PLAATSEN: {filmkeuze}: {kind} kinderen,
{volwassene} volwassenen, {student65} studenten/65+\'ers totale prijs:
€{kind*5+volwassene*8+student65*6}. Plaatsen over: {capaciteit-
(kind+volwassene+student65)}\n')
```

als er niet genoeg plaatsen zijn zegt het programma dat in de GUI voor 5 seconden en schrijft het in het logboek.

```
def resetlog():
    with open('Eindopdracht E/tickets.txt','w') as f:
        f.write(f'')
```

dit reset het logboek (wordt niet gebruikt)

```
def film_veranderd(naam):
    global filmCapaciteiten,oudefilm

    film = str(app.getOptionBox('filmkeuzes'))
    tijdslot=str(app.getOptionBox('tijdslot'))
    if oudefilm!=film:
        app.changeOptionBox('tijdslot',list(filmCapaciteiten[film].keys()))
    oudefilm=film

    tijdslot=str(app.getOptionBox('tijdslot'))

    if (film in filmCapaciteiten):
        if tijdslot in filmCapaciteiten[film]:
            app.setLabel('capaciteit', f'Nog {filmCapaciteiten[film][tijdslot]}
plaatsen beschikbaar')
    else:
        app.setLabel('capaciteit', 'Nog niets geselecteerd!')
```

als de film of tijdslot in de GUI wordt geselecteerd, verandert het programma zo dat je kan zien hoeveel plaatsen er nog vrij zijn. Als er niets geselecteerd is word dat gemeld

```
def analyseer(naam):
```

```
    global klanten,uitverkocht
    app.hideFrame('kopen')
    app.showFrame('analyse')
    klantennummers={
        'kinderen':0,
        'volwassenen':0,
        'studenten65':0,
    }
    filmnummers={'Oppenheimer':0,
        'Dune':0,
        'Avatar':0}
    for klant in klanten:
        klantennummers['kinderen']+=klant.kinderen
        klantennummers['volwassenen']+=klant.volwassenen
        klantennummers['studenten65']+=klant.studenten65
        filmnummers[klant.film]+=1
```

dit kijkt naar de klanten, wat zij allemaal hebben gekocht en analyseert de gegevens

```
    app.setLabel('groepen',f'aantal groepen: {len(klanten)}')
    app.setLabel('kinderen',f'aantal kinderen: {klantennummers["kinderen"]}')
    app.setLabel('volwassenen',f'aantal volwassenen: {klantennummers["volwassenen"]}')
    app.setLabel('studenten65',f'aantal studenten/65+\'ers:
{klantennummers["studenten65"]}')
    app.setLabel('aantalMensen',f'aantal mensen in totaal:
{int(klantennummers["kinderen"])+int(klantennummers["volwassenen"])+int(klantennummers
["studenten65"])}')
    meeste_film = max(filmnummers,key=filmnummers.get)
    app.setLabel('Oppenheimer',f'aantal Oppenheimer: {filmnummers["Oppenheimer"]},
aantal keer uitverkocht: {uitverkocht["Oppenheimer"]}')
    app.setLabel('Dune',f'aantal Dune: {filmnummers["Dune"]}, aantal keer uitverkocht:
{uitverkocht["Dune"]}')
    app.setLabel('Avatar',f'aantal Avatar: {filmnummers["Avatar"]}, aantal keer
uitverkocht: {uitverkocht["Avatar"]}')
    app.setLabel('film',f'meest gekozen film: {meeste_film}, aantal keer uitverkocht
in totaal: {uitverkocht["Oppenheimer"]+uitverkocht["Dune"]+uitverkocht["Avatar"]}')
```

laat de geanalyseerde gegevens zien in de GUI. Meeste film krijgt de film die het meest verkocht is

```
oudefilm=''



app.startFrame('kopen')
app.setBg("#c7c7c7")
app.addLabel('intro','Bioscoopticketautomaat')
```

```
app.addLabel('capaciteit',f'Nog niets geselecteerd!')

app.addOptionBox('filmkeuzes',list(filmCapaciteiten.keys()))
app.addOptionBox('tijdslot',['-tijden-'])
film_veranderd('')
app.addLabel('kindlabel','Aantal kinderen (€5)')
app.addSpinBoxRange('kind',0,128)
app.addLabel('volwassenelabel','Aantal volwassenen (€8)')
app.addSpinBoxRange('volwassene',0,128)
app.addLabel('student/65+label','Aantal studenten/65+\'ers (€6)')
app.addSpinBoxRange('student/65+',0,128)
app.addButton('koop',koop)
app.addButton('stop',lambda: app.stop())
#app.addButton('reset tickets',resetlog)
if simulerend:
    app.addButton('analyse',analyseer)
app.setOptionBoxChangeFunction('filmkeuzes', film_veranderd)
app.setOptionBoxChangeFunction('tijdslot', film_veranderd)
app.stopFrame()
```

de eerste pagina ‘kopen’ wordt geinitialiseerd met alle invoerplekken die nodig zijn en detectie van verandering van de filmkeuzes en het tijdslot. Als je simuleert is er ook een analyse-knop

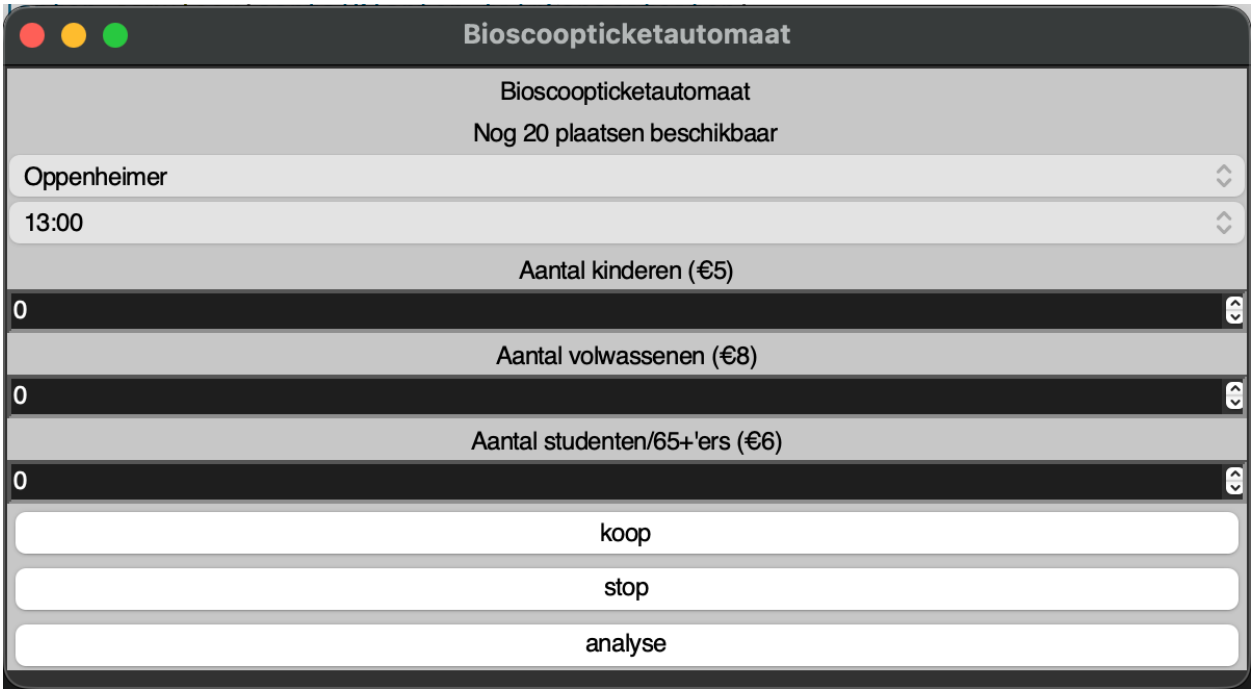
```
app.startFrame('analyse')
app.addLabel('groepen',f'aantal groepen: {0}')
app.setBg("#c7c7c7")
app.addLabel('kinderen',f'aantal kinderen: {0}')
app.addLabel('volwassenen',f'aantal volwassenen: {0}')
app.addLabel('studenten65',f'aantal studenten/65+\'ers: {0}')
app.addLabel('aantalMensen',f'aantal mensen: {0}')
app.addLabel('Oppenheimer',f'aantal Oppenheimer: {0}')
app.addLabel('Dune',f'aantal Dune: {0}')
app.addLabel('Avatar',f'aantal Avatar: {0}')
app.addLabel('film',f'meest gekozen film: {None}')
app.addButton('ga terug',lambda: (app.hideFrame('analyse'),app.showFrame('kopen')))
app.stopFrame()
app.hideFrame('analyse')
```

de tweede GUI pagina ‘analyse’ wordt geinitialiseerd en gestopt er is een knop om terug te gaan

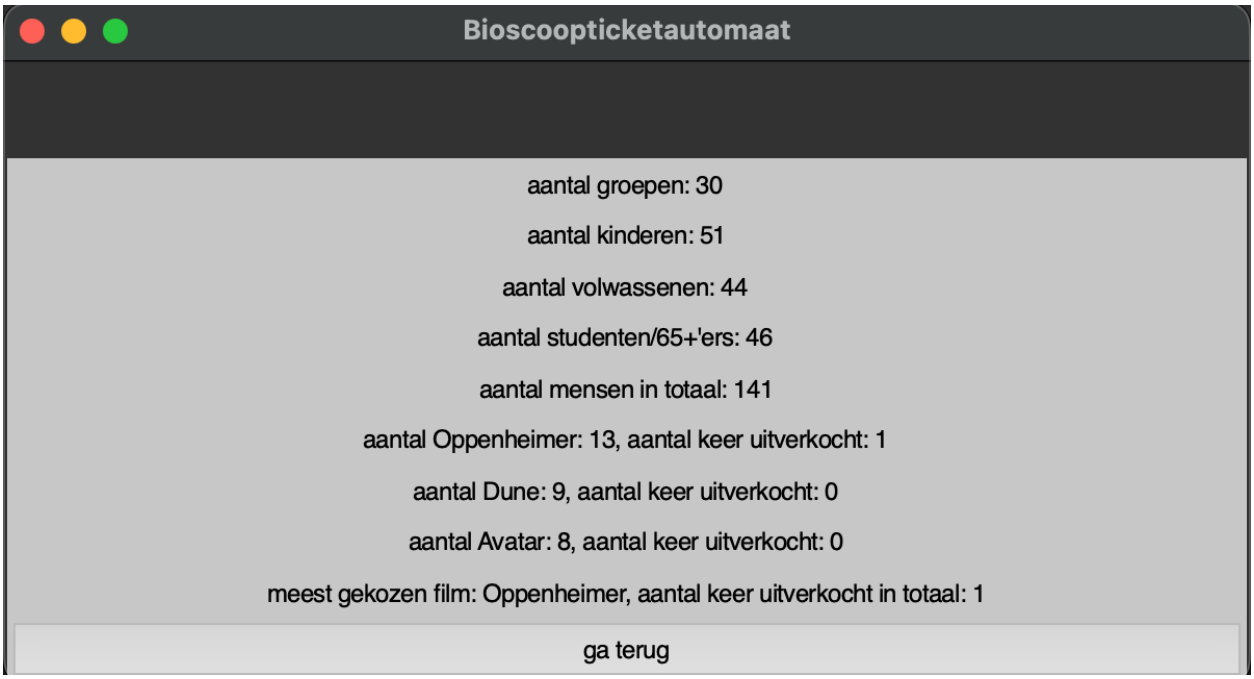
```
if simulerend:
    aantalKlanten=30
    resetlog()
    klanten=simuleer(aantalKlanten,filmCapaciteiten)
    for klant in klanten:
        koop('',True,klant)
```

```
app.go()
```
dit maakt klanten als je simulerend aan zet (bovenaan de code)

GUI

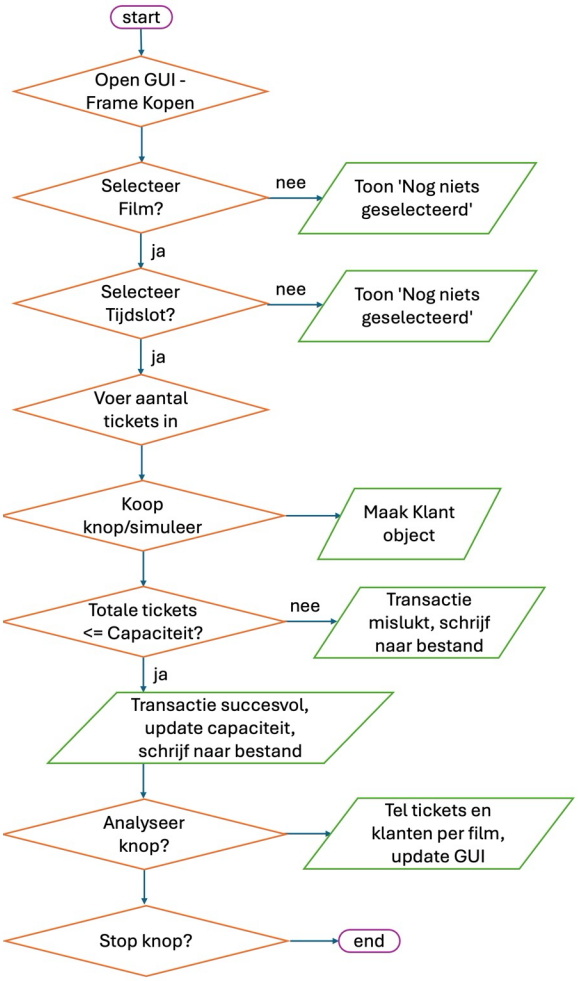


Op pagina ‘kopen’ kan je de tickets zelf kopen. Vul de gewenste film in en het gewenste tijdslot, het aantal personen dat komt, en druk op koop om iets te kopen. De knop ‘stop’ sluit het programma. De knop ‘analyse’ opent de analysepagina.



Dit is de pagina ‘analyse’. Hier kan je zien hoeveel kinderen, volwassenen en studenten/65+’ers zijn gegaan en in hoeveel groepen. Ook zie je naar welke films de groepen zijn geweest, of die uitverkocht waren en welke film het populairst is.

FSM-diagram


start
Open GUI - Frame Kopen
Selecteer Film?
nee
Toon 'Nog niets geselecteerd'
ja
Selecteer Tijdslot?
nee
Toon 'Nog niets geselecteerd'
ja
Voer aantal tickets in
Koop knop/simuleer
Maak Klant object
Totale tickets <= Capaciteit?
nee
Transactie mislukt, schrijf naar bestand
ja
Transactie succesvol, update capaciteit, schrijf naar bestand
Analyseer knop?
Tel tickets en klanten per film, update GUI
Stop knop?
end

Reflectie

Ik vond de opdrachten wel leuk om te doen als puzzel, maar zij duurden soms wel iets lang. Ik kon python namelijk al lang voordat we deze module kregen en dus had ik werkelijk niets aan de opdrachten die we voor de een of andere reden MOESTEN maken dus dat kostte me veel tijd voor niets. Het was bij de grote opdrachten wel leuk omdat dat nog steeds puzzels waren op een oké niveau de kleine gewoon niet. Het verslag maken was wel OK. Het flowdiagram iets moeilijk omdat ik nog nooit van dit concept had gehoord.

Als ik iets kon veranderen was het eerder starten met de opdracht maken, maar ik heb het nu ook afgekregen gewoon met wat stress op de laatste dag.