2019



Energy drink

Programmering - B

**Max Hansen**

Odense tekniske Gymnasium

20. september 2019

**Databaser & Gui**

Denne synopsis indeholder methoder fra programmeringsfaget

til at lave et program som har et Gui fra tkinter, og en database

fra sqlite 3

# Indledning

Et program som har en grafisk brugerflade og kan huske de data som man indtaster, selv efter programmet er lukket, er meget almindeligt i vores tid. Derfor arbejder jeg med at lave et program som har en grafisk brugerflade, som kan arbejde sammen med en database. Programmet vil komme til at omhandle energidrikke. Jeg vil i denne synopsis dokumentere min arbejdsgang med lave det her program, og komme ind på de forskellige programmeringsmetoder til planlægning og udførelse af programmet.

# Planlægning af programmet

For at kunne lave mit program skulle jeg først planlægge mit arbejde med programmet. Jeg startede med at tænke over de funktioner som mit program skulle have for at fungere.

Jeg fandt ud af at jeg gerne ville have seks funktioner i mit program. I programmet skulle brugeren af programmet kunne: Tilføje en energidrik, fjerne en energidrik, tilføje en ny producent af energidrikke, fjerne en nuværende producent af energidrikke, ændre en nuværende energidrik og ændre en nuværende producent. Nu, hvor jeg havde de funktioner som jeg vil have mit program til at have, så gik jeg i gang med at beskrive brugerhistorierne for funktionerne. Brugerhistorierne skulle jeg bruge til at kunne ”simulere”, hvordan brugeren ville kunne bruge de forskellige funktioner i programmet korrekt. Det var også for at kunne se hvordan programmet skulle sættes op i opbygningen, for at kunne give den bedst mulige oplevelse. Eftersom jeg havde seks funktioner, så ville jeg også skulle have seks brugerhistorier, da hver funktion skulle beskrives korrekt.

# Bilag

## Bilag x – Energy\_drinks\_data.py

*import* sqlite3

class Energy\_drink():

    def \_\_init\_\_(*self*, *name*, *price*, *brand*, *e\_type*):

        self.name = name

        self.price = price

        self.brand = brand

        self.e\_type = e\_type

    def set\_id(*self*,*id*):

        self.id = id

class Energy\_drink\_data():

    def \_\_init\_\_(*self*):

        self.db = sqlite3.connect('energy\_drinks.db')

    def get\_producer\_list(*self*):

        c = self.db.cursor()

        c.execute('SELECT name, location FROM producers;')

        p\_list = []

*for* p in c:

            p\_list.append((p[0], p[1]))

*return* p\_list

    def get\_type\_list(*self*):

        c = self.db.cursor()

        c.execute('SELECT types FROM drink\_types;')

        t\_list = []

*for* t in c:

            t\_list.append(t[0])

*return* t\_list

    def get\_energy\_drink\_list(*self*):

        c = self.db.cursor()

        c.execute('SELECT d.name, p.name, d.price, dd.types, d.id FROM drinks d INNER JOIN producers p ON d.producers = p.id INNER JOIN drink\_types dd ON d.type = dd.id;')

        d\_list = []

*for* d in c:

            drink = Energy\_drink(d[0], d[2], d[1], d[3])

            drink.set\_id(d[4])

            d\_list.append(drink)

*return* d\_list

    def get\_producer\_id(*self*, *p*):

        c = self.db.cursor()

        c.execute('SELECT id FROM producers WHERE name = ?;', (p.split(' ')[0],))

        p = c.fetchone()

*return* p[0]

    def get\_type\_id(*self*, *t*):

        c = self.db.cursor()

        c.execute('SELECT id FROM drink\_types WHERE types = ?;', (t,))

        e = c.fetchone()

*return* e[0]

    def add\_new\_drink(*self*, *d*):

        p = self.get\_producer\_id(d.brand)

        t = self.get\_type\_id(d.e\_type)

        c = self.db.cursor()

        c.execute('INSERT INTO drinks (name, price, producers, type) VALUES (?, ?, ?, ?);', (d.name, d.price, p, t))

        self.db.commit()

    def add\_new\_producer(*self*, *n*, *l*):

        c = self.db.cursor()

        c.execute('INSERT INTO producers (name, location) VALUES (?, ?);', (n, l))

        self.db.commit()

    def delete\_drink(*self*, *id*):

        c = self.db.cursor()

        c.execute('DELETE FROM drinks WHERE id = ?;', (id,))

        self.db.commit()

    def delete\_producer(*self*, *p*):

        d = self.get\_producer\_id(p)

        c = self.db.cursor()

        c.execute('DELETE FROM drinks WHERE producers = ?;', (d,))

        c.execute('DELETE FROM producers WHERE id = ?;', (d,))

        self.db.commit()

    def create\_tables(*self*):

*try*:

            self.db.execute("""DROP TABLE IF EXISTS drinks;""")

            self.db.execute("""DROP TABLE IF EXISTS producers;""")

            self.db.execute("""DROP TABLE IF EXISTS drink\_types;""")

            print('Table deleted')

*except* Exception *as* e:

            print('ERROR while deleting table!')

            print(e)

*try*:

            self.db.execute("""CREATE TABLE IF NOT EXISTS drinks (id INTEGER PRIMARY KEY, name TEXT, producers INTEGER, price INTEGER, type INTEGER);""")

            self.db.execute("""CREATE TABLE IF NOT EXISTS producers (id INTEGER PRIMARY KEY, name TEXT, location TEXT);""")

            self.db.execute("""CREATE TABLE IF NOT EXISTS drink\_types (id INTEGER PRIMARY KEY, types TEXT);""")

            print('Tables created succesfully')

*except* Exception *as* e:

            print('ERROR while creating tables!')

            print(e)

        self.db.execute("""INSERT INTO drinks (name, producers, price, type) VALUES (?, ?, ?, ?);""", ('Monster Energy', 1, 20, 1))

        self.db.execute("""INSERT INTO producers (name, location) VALUES (?, ?);""", ('Monster\_Energy', 'USA'))

        self.db.execute("""INSERT INTO drink\_types (types) VALUES (?);""", ("Standard",))

        self.db.execute("""INSERT INTO drink\_types (types) VALUES (?);""", ("Sukkerfri",))

        self.db.commit()

## Bilag x – Energy\_drinks\_gui.py

*from* Energy\_drinks\_data *import* Energy\_drink, Energy\_drink\_data

*import* tkinter *as* tk

*import* tkinter.ttk *as* ttk

class Energy\_drink\_gui(ttk.Frame):

    def \_\_init\_\_(*self*, *master* = None):

        ttk.Frame.\_\_init\_\_(self, master)

        self.data = Energy\_drink\_data()

        self.build\_GUI()

        self.update\_label()

    def update\_label(*self*):

        l = self.data.get\_energy\_drink\_list()

        self.drinks\_label.config(*text* = 'Der er {} registrede energidrikke i databasen'.format(len(l)))

        self.db\_view.delete(\*self.db\_view.get\_children())

*for* e in l:

            self.db\_view.insert("", tk.END, *values* = (e.name, e.brand, e.price, e.e\_type, e.id))

    def on\_drink\_selected(*self*, *event*):

        cur\_item = self.db\_view.item(self.db\_view.focus())['values']

*if* len(cur\_item) > 0:

            self.label\_current\_name.config(*text* = 'Navn: {}'.format(cur\_item[0]))

            self.label\_current\_producer.config(*text* = 'Producent: {}'.format(cur\_item[1]))

            self.label\_current\_price.config(*text* = 'Pris: {}'.format(cur\_item[2]))

            self.label\_current\_type.config(*text* = 'Type: {}'.format(cur\_item[3]))

    def clear\_drink\_entry(*self*):

        self.entry\_name.delete(0, tk.END)

        self.entry\_price.delete(0, tk.END)

        self.cb\_producers.set('')

        self.cb\_type.set('')

    def clear\_producer\_entry(*self*):

        self.entry\_producer\_name.delete(0, tk.END)

        self.entry\_producer\_location.delete(0, tk.END)

    def clear\_current\_drink\_label(*self*):

        self.label\_current\_name.configure(*text* = 'Navn: ')

        self.label\_current\_producer.configure(*text* = 'Producent: ')

        self.label\_current\_price.configure(*text* = 'Price: ')

        self.label\_current\_type.configure(*text* = 'Type: ')

    def add\_new\_drink(*self*):

        e = Energy\_drink(self.entry\_name.get(), int(self.entry\_price.get()), self.cb\_producers.get(), self.cb\_type.get())

        self.data.add\_new\_drink(e)

        self.update\_label()

        self.clear\_drink\_entry()

    def add\_new\_producer(*self*):

        self.data.add\_new\_producer(self.entry\_producer\_name.get(), self.entry\_producer\_location.get())

        producers = self.data.get\_producer\_list()

        self.cb\_producers = ttk.Combobox(self.button\_panel, *values* = producers, *state* = 'readonly')

        self.cb\_producers.grid(*row* = 2, *column* = 2, *pady* = (0,5))

        self.cb\_delete\_producers = ttk.Combobox(self.button\_panel, *values* = producers, *state* = 'readonly')

        self.cb\_delete\_producers.grid(*row* = 9, *column* = 2)

        self.clear\_producer\_entry()

    def delete\_current\_drink(*self*):

        cur\_item = self.db\_view.focus()

*if* len(self.db\_view.item(cur\_item)['values']) >=4:

            self.data.delete\_drink(self.db\_view.item(cur\_item)['values'][4])

        self.update\_label()

        self.clear\_current\_drink\_label()

    def delete\_producer(*self*):

        self.data.delete\_producer(self.cb\_delete\_producers.get())

        self.update\_label()

        producers = self.data.get\_producer\_list()

        self.cb\_producers = ttk.Combobox(self.button\_panel, *values* = producers, *state* = 'readonly')

        self.cb\_producers.grid(*row* = 2, *column* = 2, *pady* = (0,5))

        self.cb\_delete\_producers = ttk.Combobox(self.button\_panel, *values* = producers, *state* = 'readonly')

        self.cb\_delete\_producers.grid(*row* = 9, *column* = 2)

        self.cb\_delete\_producers.set('')

    def build\_GUI(*self*):

        self.data\_panel = ttk.Frame(self)

        self.button\_panel = ttk.Frame(self)

        self.button\_panel.grid\_columnconfigure(3, *minsize* = 200)

        self.drinks\_label = ttk.Label(self.button\_panel, *text* = 'Der er {} registrede energidrikke i databasen'.format(None))

        self.drinks\_label.grid(*row* = 0, *column* = 0)

        self.button\_update = ttk.Button(self.button\_panel, *text* = 'Opdater', *command* = self.update\_label)

        self.button\_update.grid(*row* = 1, *column* = 0)

        self.label\_name = ttk.Label(self.button\_panel, *text* = 'Navn')

        self.label\_name.grid(*row* = 0, *column* = 1)

        self.entry\_name = ttk.Entry(self.button\_panel, *width* = 23)

        self.entry\_name.grid(*row* = 0, *column* = 2, *pady* = (0,5))

        self.label\_price = ttk.Label(self.button\_panel, *text* = 'Pris')

        self.label\_price.grid(*row* = 1, *column* = 1)

        self.entry\_price = ttk.Entry(self.button\_panel, *width* = 23)

        self.entry\_price.grid(*row* = 1, *column* = 2, *pady* = (0,5))

        producers = self.data.get\_producer\_list()

        label\_producers = ttk.Label(self.button\_panel, *text* = 'Producenter')

        label\_producers.grid(*row* = 2, *column* = 1)

        self.cb\_producers = ttk.Combobox(self.button\_panel, *values* = producers, *state* = 'readonly')

        self.cb\_producers.grid(*row* = 2, *column* = 2, *pady* = (0,5))

        types = self.data.get\_type\_list()

        label\_type = ttk.Label(self.button\_panel, *text* = 'Type')

        label\_type.grid(*row* = 3, *column* = 1)

        self.cb\_type = ttk.Combobox(self.button\_panel, *values* = types, *state* = 'readonly')

        self.cb\_type.grid(*row* = 3, *column* = 2, *pady* = (0,5))

        self.button\_add = ttk.Button(self.button\_panel, *text* = 'Tilføj energidrik', *command* = self.add\_new\_drink)

        self.button\_add.grid(*row* = 4, *column* = 2, *pady* = (0,10))

        self.label\_producer\_name = ttk.Label(self.button\_panel, *text* = 'Producentens navn')

        self.label\_producer\_name.grid(*row* = 6, *column* = 1)

        self.entry\_producer\_name = ttk.Entry(self.button\_panel, *width* = 23)

        self.entry\_producer\_name.grid(*row* = 6, *column* = 2, *pady* = (0,5))

        self.label\_producer\_location = ttk.Label(self.button\_panel, *text* = 'Producentens lokation')

        self.label\_producer\_location.grid(*row* = 7, *column* = 1, *padx* = (0,17))

        self.entry\_producer\_location = ttk.Entry(self.button\_panel, *width* = 23)

        self.entry\_producer\_location.grid(*row* = 7, *column* = 2, *pady* = (0,5))

        self.button\_add\_producer = ttk.Button(self.button\_panel, *text* = 'Tilføj producent', *command* = self.add\_new\_producer)

        self.button\_add\_producer.grid(*row* = 8, *column* = 2, *pady* = (0,20))

        self.label\_current\_name = ttk.Label(self.button\_panel, *text* = 'Navn: ')

        self.label\_current\_name.grid(*row* = 0, *column* = 3)

        self.label\_current\_producer = ttk.Label(self.button\_panel, *text* = 'Producent: ')

        self.label\_current\_producer.grid(*row* = 1, *column* = 3)

        self.label\_current\_price = ttk.Label(self.button\_panel, *text* = 'Pris: ')

        self.label\_current\_price.grid(*row* = 2, *column* = 3)

        self.label\_current\_type = ttk.Label(self.button\_panel, *text* = 'Type: ')

        self.label\_current\_type.grid(*row* = 3, *column* = 3)

        self.button\_delete = ttk.Button(self.button\_panel, *text* = 'Fjern energidrik', *command* = self.delete\_current\_drink)

        self.button\_delete.grid(*row* = 4, *column* = 3)

        self.label\_producer = ttk.Label(self.button\_panel, *text* = 'Producent')

        self.label\_producer.grid(*row* = 9, *column* = 1)

        self.cb\_delete\_producers = ttk.Combobox(self.button\_panel, *values* = producers, *state* = 'readonly')

        self.cb\_delete\_producers.grid(*row* = 9, *column* = 2)

        self.button\_delete\_producer = ttk.Button(self.button\_panel, *text* = 'Fjern producent', *command* = self.delete\_producer)

        self.button\_delete\_producer.grid(*row* = 10, *column* = 2)

        self.db\_view = ttk.Treeview(self.data\_panel, *column* = ('column1', 'column2', 'column3', 'column4', 'column5'), *show* = 'headings', *height* = 49)

        self.db\_view.bind("<ButtonRelease-1>", self.on\_drink\_selected)

        self.db\_view.heading('#1', *text* = 'Navn')

        self.db\_view.heading('#2', *text* = 'Producent')

        self.db\_view.heading('#3', *text* = 'Pris')

        self.db\_view.heading('#4', *text* = 'Type')

        self.db\_view.heading('#5', *text* = 'id')

        self.db\_view['displaycolumns'] = ('column1', 'column2', 'column3', 'column4')

        scroll\_bar\_y = ttk.Scrollbar(self.data\_panel, *command* = self.db\_view.yview, *orient* = tk.VERTICAL)

        self.db\_view.configure(*yscrollcommand* = scroll\_bar\_y.set)

        self.db\_view.pack(*side* = tk.RIGHT)

        self.data\_panel.pack(*side* = tk.RIGHT)

        self.button\_panel.pack(*side* = tk.TOP)

        self.pack()

root = tk.Tk()

root.iconbitmap('./Icon/icon\_AQK\_icon.ico')

root.geometry('1280x720')

root.state('zoomed')

app = Energy\_drink\_gui(root)

app.master.title('Energidrikke')

app.mainloop()

## Bilag x – Brugerhistorier og iterationer

### Brugerhistorie 1: Tilføj energidrik

#### Planer til iteration 1

Denne brugerhistorie kan startes af brugeren, uanset hvilken side brugeren er på.

1. Brugeren vælger at tilføje en ny energidrik.
2. Brugeren indtaster oplysninger til programmet:
   1. Energidrikkens navn
   2. Energidrikkens producent
   3. Energidrikkens pris
   4. Energidrikkens type
3. Brugeren trykker på ”Tilføj energidrik” for at tilføje en energidrik med de oplysninger som brugeren har indtastet.
4. Programmet tilføjer energidrikken i databasen.
5. Programmet opdaterer den viste liste med energidrikke.
6. Programmet lægger 1 til antallet af energidrikke i databasen.
7. Programmet fjerner alle brugerens input, så det er klar til noget nyt.

#### Planer til iteration 2

1. Brugeren vælger at tilføje en ny energidrik.
2. Brugeren indtaster oplysninger til programmet:
   1. Energidrikkens navn
   2. Energidrikkens producent
   3. Energidrikkens pris
   4. Energidrikkens type
3. Brugeren trykker på ”Tilføj energidrik” for at tilføje en energidrik med de oplysninger som brugeren har indtastet.
   1. Programmet tjekker om brugeren har indtastet et tal som pris.
      1. Hvis pris ikke er et tal, programmet laver en fejl og der kommer en rød firkant på skærmen med beskeden ”Prisen skal være et tal!”.
4. Programmet tilføjer energidrikken i databasen.
5. Programmet opdaterer den viste liste med energidrikke.
6. Programmet lægger 1 til antallet af energidrikke i databasen.
7. Programmet fjerner alle brugerens input, så det er klar til noget nyt.
8. Programmet viser en grøn firkant på skærmen med beskeden ”Energidrikken blev tilføjet til databasen!”.

### Brugerhistorie 2: Slet energidrik

#### Planer til iteration 1

Denne brugerhistorie kan startes af brugeren, uanset hvilken side brugeren er på.

1. Brugeren vælger den energidrik der skal slettes fra listen over energidrikke.
2. Brugeren trykker på ”Slet energidrik”.
3. Programmet sletter den valgte energidrik fra databasen.
4. Programmet trækker 1 fra antallet af energidrikke i databasen.

#### Planer til iteration 2

1. Brugeren vælger den energidrik der skal slettes fra listen over energidrikke.
2. Brugeren trykker på ”Slet energidrik”.
3. Programmet sletter den valgte energidrik fra databasen.
4. Programmet trækker 1 fra antallet af energidrikke i databasen.
5. Programmet laver en grøn firkant på skærmen med beskeden ”Energidrikken blev fjernet fra databasen!”.

### Brugerhistorie 3: Tilføj producent

#### Planer til iteration 1

Denne brugerhistorie kan startes af brugeren, uanset hvilken side brugeren er på.

1. Brugeren vælger at tilføje en ny producent
2. Brugeren indtaster oplysninger til programmet:
   1. Producentens navn
   2. Producentens lokation
3. Brugeren trykker på ”Tilføj producent” for at tilføje en producent med de oplysninger som brugeren har indtastet.
4. Programmet tilføjer producenten til databasen over producenter.
5. Programmet fjerner alle brugerens input, så det er klar til noget nyt.

#### Planer til iteration 2

1. Brugeren vælger at tilføje en ny producent
2. Brugeren indtaster oplysninger til programmet:
   1. Producentens navn
   2. Producentens lokation
3. Brugeren trykker på ”Tilføj producent” for at tilføje en producent med de oplysninger som brugeren har indtastet.
4. Programmet tilføjer producenten til databasen over producenter.
5. Programmet fjerner alle brugerens input, så det er klar til noget nyt.
6. Programmet laver en grøn firkant på skærmen med beskeden ”Producenten blev tilføjet til databasen!”.

### Brugerhistorie 4: Fjern producent

#### Plan til iteration 1

Denne brugerhistorie kan startes af brugeren, uanset hvilken side brugeren er på.

1. Brugeren vælger en producent fra en dropdown menu som brugeren vil slette.
2. Brugeren trykker derefter på ”Slet producent”.
3. Programmet sletter den valgte producent fra databasen over producenter.
4. Programmet sletter også energidrikke med den valgte producent, fra databasen over energidrikke.
5. Programmet fjerner brugerens input fra dropdown menuen, så det er klar til noget nyt.

#### Plan til iteration 2

1. Brugeren vælger en producent fra en dropdown menu som brugeren vil slette.
2. Brugeren trykker derefter på ”Slet producent”.
3. Programmet sletter den valgte producent fra databasen over producenter.
4. Programmet sletter også energidrikke med den valgte producent, fra databasen over energidrikke.
5. Programmet fjerner brugerens input fra dropdown menuen, så det er klar til noget nyt.
6. Programmet laver en grøn firkant på skærmen med beskeden ”Producenten blev fjernet fra databasen!”.

### Brugerhistorie 5: Ændre energidrik

Denne brugerhistorie kan startes af brugeren, uanset hvilken side brugeren er på.

1. Brugeren vælger en energidrik fra listen over energidrikke.
2. Brugeren trykker på ”Ændre”.
3. Brugeren ændrer de oplysninger som skal ændres.
4. Brugeren trykker på ”Ændre energidrik”.
5. Programmet opdaterer energidrikken i databasen.
6. Programmet opdaterer listen med energidrikke, så oplysningerne er korrekte.
7. Programmet fjerner inputstederne til at ændre energidrikke, så det er klar til nyt input.

### Brugerhistorie 6: Ændre producent

Denne brugerhistorie kan startes af brugeren, uanset hvilken side brugeren er på.

1. Brugeren vælger en producent fra en dropdown menu som brugeren vil ændre.
2. Brugeren trykker på ”Ændre”.
3. Brugeren ændrer de oplysninger som skal ændres.
4. Brugeren trykker på ”Ændre producent”.
5. Programmet opdaterer producenten i databasen over producenter.
6. Programmet opdaterer listen med energidrikke, så oplysningerne er korrekte.
7. Programmet fjerner inputstederne til at ændre producenten, samt nulstiller dropdown menuen, så det er klar til noget nyt.

### Iteration 1:

#### Brugerhistorier:

* Brugerhistorie 1, Brugerhistorie 2, Brugerhistorie 3 og Brugerhistorie 4.

### Iteration 2:

#### Brugerhistorier:

* Brugerhistorie 1, Brugerhistorie 2, Brugerhistorie 3 og Brugerhistorie 4.

### Iteration 3:

#### Brugerhistorier:

* Brugerhistorie 1, Brugerhistorie 2, Brugerhistorie 3, Brugerhistorie 4, Brugerhistorie 5 og Brugerhistorie 6.