```python
import sqlalchemy as sql
from sqlalchemy.exc import OperationalError as sqlOpErr
from Lib.Data_import import DataToImport


class DataToDatabase(DataToImport):     # erbt Dataimport Klasse

    def __init__(self):

        self.rows = 0

        super().__init__()

    def create_table(self, tablename, di, connection):
        """
        Die Funktion erzeugt eine Tabelle in der Datenank
        :param tablename: übergebener Tabellenname
        :param di: Inhalt der Tabelle
        :param connection: Verbindung zur Datenbank
        kann die Tabelle nicht erzeugt werden wird eine Exception ausgelöst
        """
        # text fuer Create bauen (x float, y0 float, y1 float, ... )
        if len(di.x) > 0:
            text_create = "(" + di.x[0] + " float, "
            for i in range(di.Anzahl_Spalten - 1):
                text_create += di.y[i + 1][0] + " float, "
            if tablename == 'test':
                text_create = text_create[:-7]
                text_create += "string, "
            text_create = text_create[:-2]
            text_create += ")"
        else:
            print("Array ist leer. kann nicht ausgeführt werden")
```

```
34              pass
35
36          rowsintable = 0     # Tabelle als leer
   vorbelegen
37
38          try:
39              connection.execute(sql.text("CREATE TABLE
    " + tablename + " " + text_create))   # Tabelle
   einfügen
40          except sqlOpErr:
41              print("Tabelle existiert schon")
42              result = connection.execute(sql.text("
   SELECT * FROM " + tablename))
43              for row in result:
44                  rowsintable += 1  # i > 0 → Tabelle
   ist schon befüllt   # prüfen, ob Tabelle befüllt ist
   , wenn sie existiert
45          self.rows = rowsintable
46
47      def create_table_id4(self, tablename, di,
   connection):
48          """
49          Die Funktin erzeugt die Tabelle für die 4
   idealen Funktionen,
50          nötig weil das Format anders ist als oben
51          :param tablename: übergebener Tabellenname
52          :param di: Inhalt für die Tabelle
53          :param connection: Verbindung zur Datenbank
54          """
55          # text fuer Create bauen (x float, y0 float,
   y1 float, ... )
56          text_create = "(" + di.y[1][0][0] + " float
   , "
57          for i in range(1, di.Anzahl_Spalten):
58              text_create += di.y[i][3][2] + " float, "
59          text_create = text_create[:-2]
60          text_create += ")"
61
62          rowsintable = 0 # Tabelle als leer vorbelegen
63
64          try:
```

```
65              connection.execute(sql.text("CREATE
   TABLE " + tablename + " " + text_create))
66          except sqlOpErr:
67              print("Tabelle existiert schon")
68              result = connection.execute(sql.text("
   SELECT * FROM " + tablename))
69              for row in result:
70                  rowsintable += 1  # i > 0 → Tabelle
   ist schon befüllt
71          self.rows = rowsintable
72
73
74      def data_to_table(self, tablename, di,
   connection):
75          """
76          Funktion befüllt die Tabelle
77          :param tablename: übergebener Tabellenname
78          :param di: Inhalt für die Tabelle
79          :param connection: Verbindung zur Datenbank
80          """
81          rowsintable = self.rows
82
83          if rowsintable == 0:
84              # TEXT_INSERT bauen (x, y1, y2, ...)
85              text_insert = "(" + di.x[0] + ", "
86              for i in range(di.Anzahl_Spalten - 1):
87                  text_insert += di.y[i + 1][0] + ", "
88              text_insert = text_insert[:-2]
89              text_insert += ")"
90
91              # TEXT VALUES bauen VARIABEL  (:x, :y1
   , :y2, ...)
92              text_values = "(:" + di.x[0] + ", :"
93              for i in range(di.Anzahl_Spalten - 1):
94                  text_values += di.y[i + 1][0] +
   ", :"
95              text_values = text_values[:-3]
96              text_values += ")"
97
98              for j in range(1, len(di.x)):  # j steht
    für die Zeilen
```

```
 99                 dict2 = {di.x[0]: di.x[j]}
100                 for i in range(1, di.Anzahl_Spalten
    ):  # i steht für die Spalten
101                     dict2.update({di.y[i][0]: di.y[i
    ][j]})  # dictionary bauen
102                 connection.execute(sql.text("INSERT
    INTO " + tablename + " " + text_insert + " VALUES "
     + text_values), [dict2])
103             connection.commit()
104         else:
105             print("Tabelle bereits befüllt")
106
107
108
109     def data_to_table_id4(self, tablename, di,
    connection):
110         """
111         Funktion befüllt die Tabelle der 4 idealen
    Funktionen
112         :param tablename: übergebener Tabellenname
113         :param di: Inhalt für die Tabelle
114         :param connection: Verbindung zur Datenbank
115         """
116         rowsintable = self.rows
117
118         if rowsintable == 0:
119             ####### TEXT_INSERT bauen (x, y1, y2
    , ...)
120             text_insert = "(" + di.y[1][0][0] + ", "
        # 'x'
121             # for i in range(di.Anzahl_Spalten - 2):
122             for i in range(1, di.Anzahl_Spalten
    ):      # 'ynr' Nummer der Funktion, z.B. y36
123                 text_insert += di.y[i][1][0] + ", "
124             text_insert = text_insert[:-2]
125             text_insert += ")"
126
127             ####### TEXT VALUES bauen VARIABEL  (:x
    , :y1, :y2, ...)
128             text_values = "(:" + di.y[1][0][0] +
    ", :"
```

```
129             # for i in range(di.Anzahl_Spalten - 2):
130             for i in range(1, di.Anzahl_Spalten):
131                 text_values += di.y[i][1][0] + ", :"
132             text_values = text_values[:-3]
133             text_values += ")"
134 
135             for j in range(1, len(di.y[1][0])):  # j
     steht für die Zeilen
136                 dict2 = {di.y[1][0][0]: di.y[1][0][j
    ]}      # 'x: x-Wert'
137                 for i in range(1, di.Anzahl_Spalten
    ):  # i steht für die Spalten; 4 Spalten für die 4
    idealen Funktionen -
138                     dict2.update({di.y[i][1][0]: di.
    y[i][1][j]})  # dictionary zeilenweise bauen
139                 connection.execute(sql.text("INSERT
    INTO " + tablename + " " + text_insert + " VALUES "
     + text_values), [dict2])  # an Datenbank senden
140             connection.commit()
141         else:
142             print("Tabelle bereits befüllt")
143 
144 
```