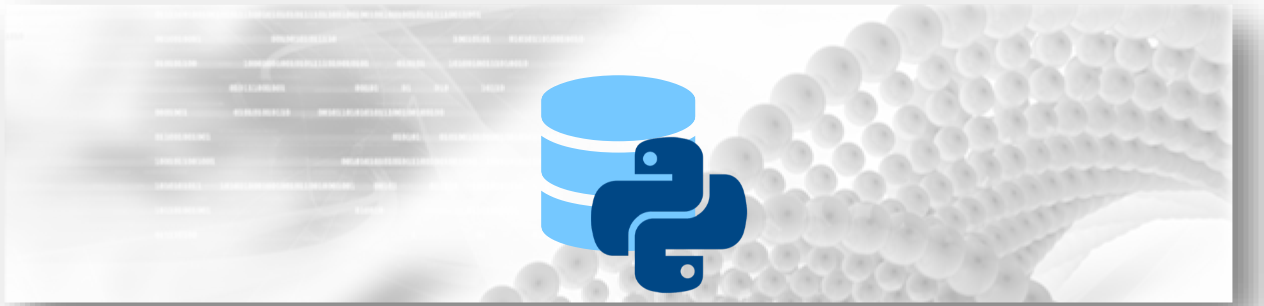
ACLS

Databases



SS 2020 – Week 6
April 23

Schedule

Calendar Week	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Module Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Date	17.02.	24.02.	02.03.	09.03.	16.03.	23.03.	30.03.	06.04.	13.04.	20.04.	27.04.	04.05.	11.05.	18.05.
Topic											NoSQL			

?

February March April May

Introduction & RDB	Data Warehouse
Structured Query Language	Not only SQL
Database & Python	Graph Database
Database & R	Special

Content

• Review		5’
• SQLite - rowid	🎓	20’
• sqlite3 - fetch	🎓	20’
• Interface Module	🎓	45’
• User Interface	🎓	45’
• Exercises	🔧	90’

Content

• Review		5’
• SQLite - rowid	🎓	20’
• sqlite3 - fetch	🎓	20’
• Interface Module	🎓	45’
• User Interface	🎓	45’
• Exercises	🔧	90’

Review (1/2)

Python - MySQL

- **Connect**
- Create
- Insert
- Query

```
# import
import mysql.connector

# connect
myConn = mysql.connector.connect(user='user',
                                 password='password',
                                 host='server')


# cursor
myCursor = myConn.cursor()
```

Review (1/2)

Python - MySQL

- Connect
- **Create**
- Insert
- Query

```
# create
myCursor.execute('CREATE DATABASE databasename')
myCursor.execute('CREATE TABLE t1 (id int NOT NULL AUTO_INCREMENT,
                                   p1 float,
                                   p2 VARCHAR(20),
                                   PRIMARY KEY (id))')
```

Review (1/2)

Python - MySQL

- Connect
- Create
- **Insert**
- Query

```
# insert
statement = ('INSERT INTO t1 (p1, p2) VALUES (%s, %s)')
values = (12.34, 'shrubbery')
myCursor.execute(statement, values)
myConn.commit()
```

Review (1/2)

Python - MySQL

- Connect
- Create
- Insert
- **Query**

```
# query
myCursor.execute('SELECT * FROM t1')
records = myCursor.fetchall()
```

Review (2/2)

Python - SQLite

- Connect
- Create
- Insert
- Query

```
# import
import sqlite3

# connect
myConn = sqlite3.connect('databasefile')

# cursor
myCursor = myConn.cursor()

# create
myCursor.execute('CREATE TABLE t1 (id integer NOT NULL PRIMARY KEY,
                                   p1 float,
                                   p2 VARCHAR(20)')


# insert
statement = ('INSERT INTO t1 (p1, p2) VALUES (?, ?)')
values = (12.34, 'shrubbery')
myCursor.execute(statement, values)
myConn.commit()

# query
myCursor.execute('SELECT * FROM t1')
records = myCursor.fetchall()
```

Content

SQLite

rowid

By default, all rows within SQLite tables have a 64-bit signed integer key that uniquely identifies the row within its table. This integer is usually called the "**rowid**".

rowid	first_name	last_name	Birth_date
1	Tom	Smith	1967-03-06
2	Jane	Jones	1978-11-27
3	Will	Thomsen	1983-06-13

⚠ no SQL standard

SQLite

rowid

An attribute defined with the keywords "INTEGER PRIMARY KEY" automatically becomes an alias for the rowid.

```
CREATE TABLE IF NOT EXISTS persons(
                    id INTEGER PRIMARY KEY,
                    first_name VARCHAR(50),
                    last_name VARCHAR(50),
                    birth_date DATE
                    );
```

rowid	id	first_name	last_name	Birth_date
1	1	Tom	Smith	1967-03-06
2	2	Jane	Jones	1978-11-27
3	3	Will	Thomsen	1983-06-13

SQLite

rowid

To retrieve the rowid, it has to be explicitly stated in the query:

```
SELECT rowid,* FROM persons;
```

The rowid can be used in conditions as any other attribute:

```
SELECT * FROM persons WHERE rowid=3;
```

SQLite provides a function to easily retrieve the rowid of the last inserted record:

```
SELECT last_insert_rowid();
```

Python

```
indentifier = myCursor.lastrowid
```

SQLite

WITHOUT ROWID

Only tables defined as WITHOUT ROWID tables don’t have a rowid:

```
CREATE TABLE IF NOT EXISTS wordcount(
  word TEXT PRIMARY KEY,
  cnt INTEGER
) WITHOUT ROWID;
```

Content

•	Review		5’
•	SQLite - rowid	🎓	20’
•	sqlite3 - fetch	🎓	20’
•	Interface Module	🎓	45’
•	User Interface	🎓	45’
•	Exercises	🔧	90’

sqlite3

fetch

Fetch **all** records of the resulting table of a query:

```
myCursor.execute(statement,values)
records = myCursor.fetchall()
```

Fetch the **next record** of the resulting table of a query :

```
myCursor.execute(statement,values)
record = myCursor.fetchone()
```

Fetch the **next n records** of the resulting table of a query :

```
myCursor.execute(statement,values)
records = myCursor.fetchmany(n)
```

sqlite3

no need for fetch?

sqlite3 also allows to access rows (records) directly through the cursor object:

```
myCursor.execute(statement,values)
for row in myCursor:
        do something with the data
```

Advantages

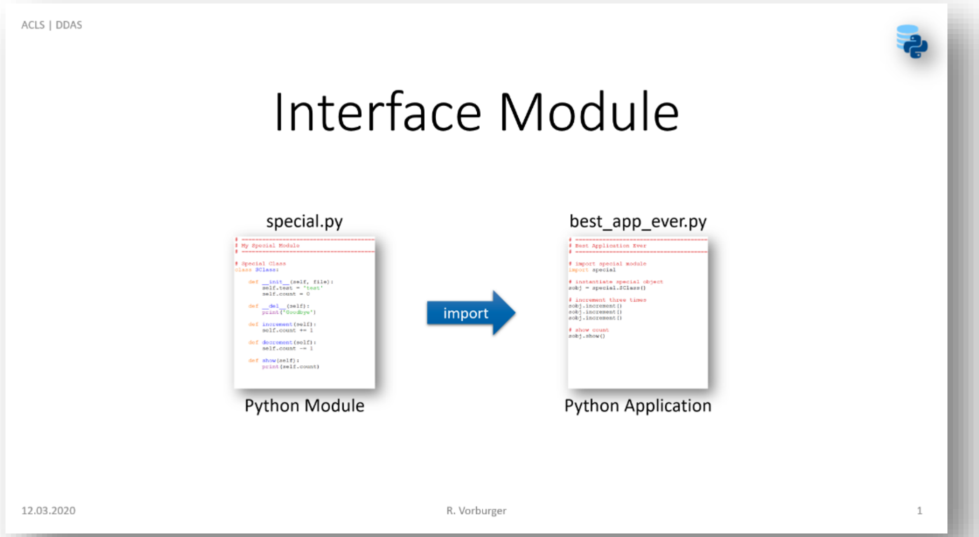
- easy implementation
- less memory used

Disadvantages

- open connection to database needed
- not conform with other database modules (e.g. MySQL Connector)

Content

• Review		5’
• SQLite - rowid	🎓	20’
• sqlite3 - fetch	🎓	20’
• **Interface Module**	🎓	**45’**
• User Interface	🎓	45’
• Exercises	🔧	90’



ACLS | DDAS

Interface Module

special.py

```
# ==========================================
# My Special Module
# ==========================================

# Special Class
class SClass:

    def __init__(self, file):
        self.test = 'test'
        self.count = 0

    def __del__(self):
        print('Goodbye')

    def increment(self):
        self.count += 1

    def decrement(self):
        self.count -= 1

    def show(self):
        print(self.count)
```

Python Module

best_app_ever.py

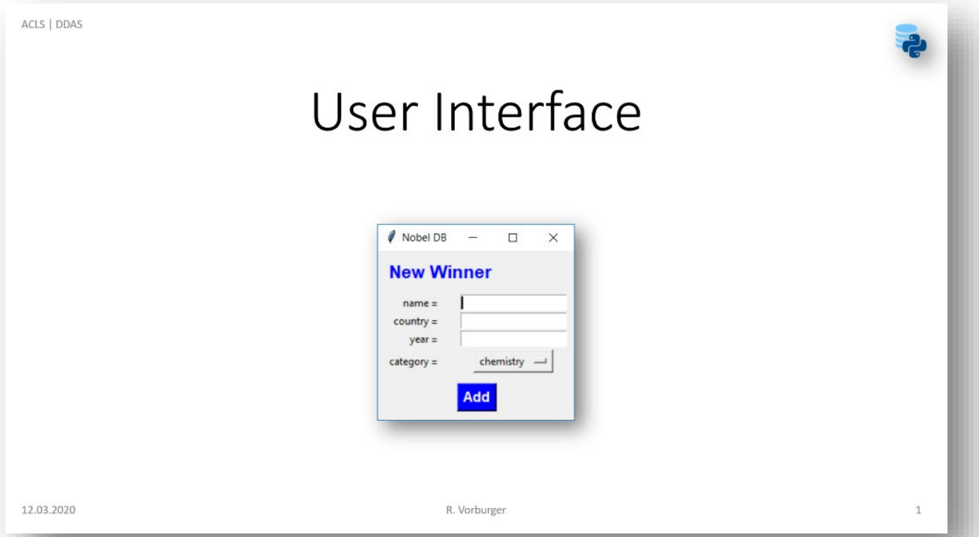
```
# ==========================================
# Best Application Ever
# ==========================================

# import special module
import special

# instantiate special object
sobj = special.SClass()

# increment three times
sobj.increment()
sobj.increment()
sobj.increment()

# show count
sobj.show()
```

Python Application

12.03.2020 R. Vorburger 1

Content

• Review		5’
• SQLite - rowid	🎓	20’
• sqlite3 - fetch	🎓	20’
• Interface Module	🎓	45’
• **User Interface**	🎓	**45’**
• Exercises	🎓	90’


ACLS | DDAS

User Interface

Nobel DB

New Winner

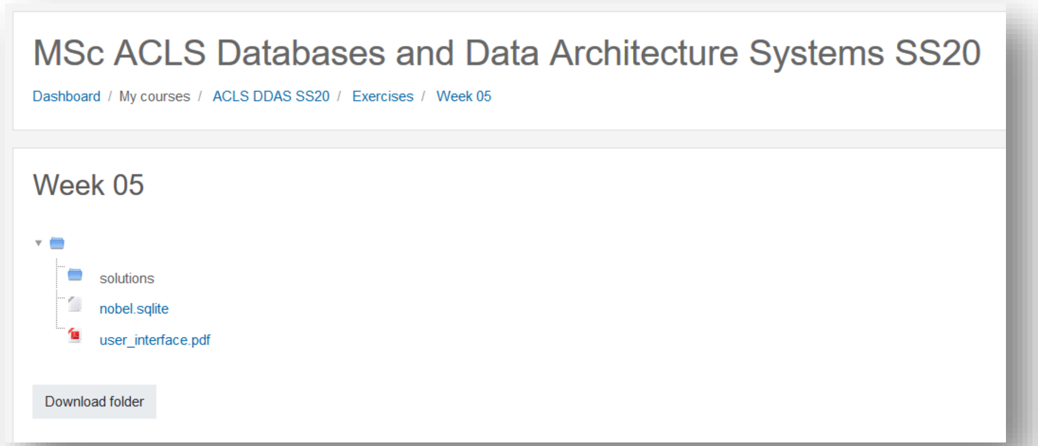
name =
country =
year =
category = chemistry

Add

12.03.2020 R. Vorburger 1

Content

• Review		5’
• SQLite - rowid	🎓	20’
• sqlite3 - fetch	🎓	20’
• Interface Module	🎓	45’
• User Interface	🎓	45’
• **Exercises**	🔧	90’


MSc ACLS Databases and Data Architecture Systems SS20

Dashboard / My courses / ACLS DDAS SS20 / Exercises / Week 05

Week 05

- solutions
- nobel.sqlite
- user_interface.pdf

Download folder