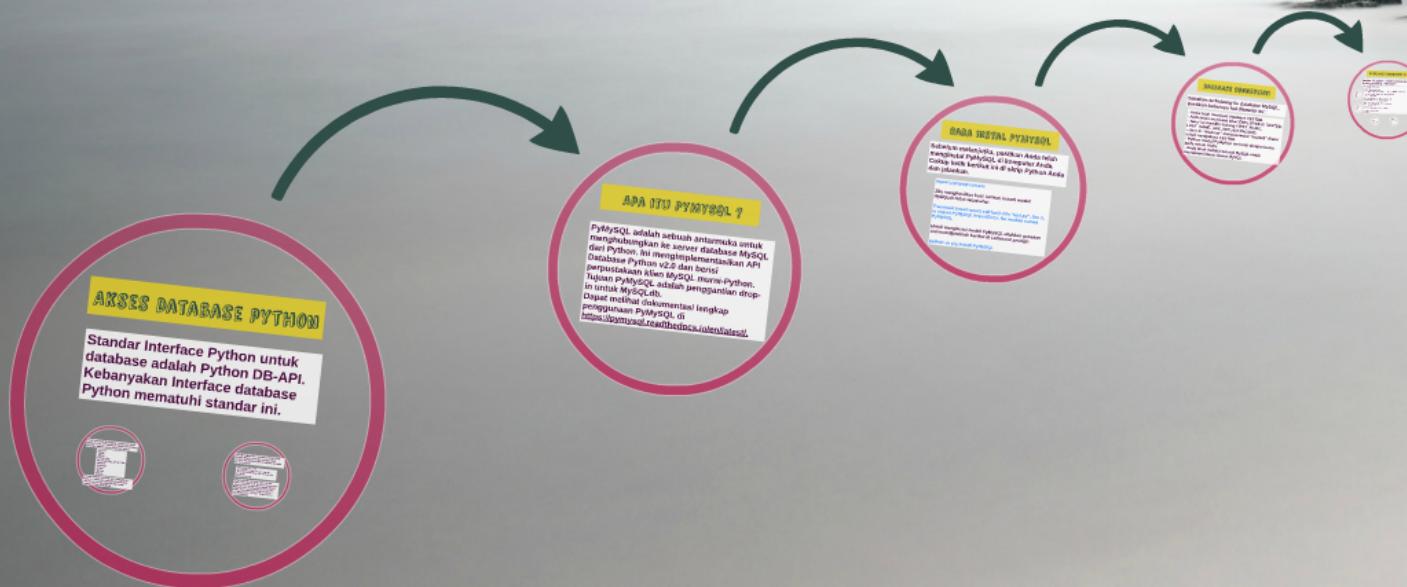
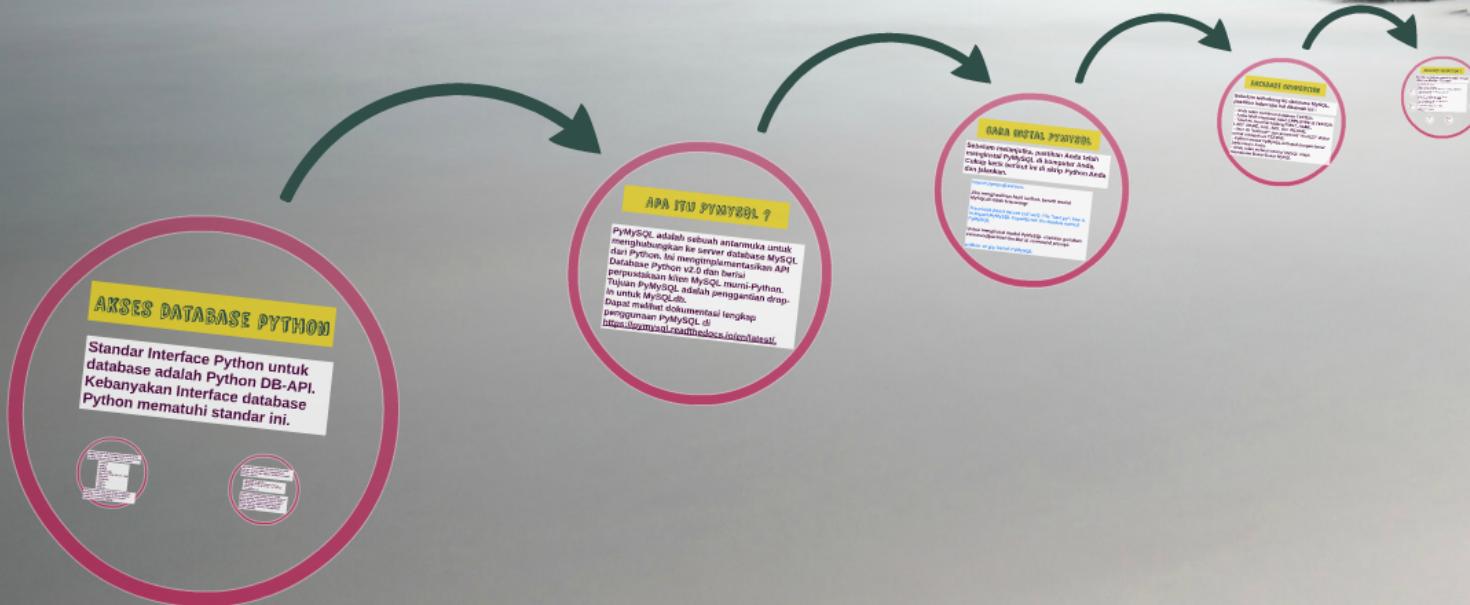


13. Pemrograman Berorientasi Objek



13. Pemrograman Berorientasi Objek



AKSES DATABASE PYTHON

Standar Interface Python untuk database adalah Python DB-API. Kebanyakan Interface database Python mematuhi standar ini.

Anda bisa memilih database yang tepat untuk aplikasi Anda. API Database Python mendukung berbagai macam server database seperti:

- Oracle
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase
- DB2

Selagi contohnya, jika Anda perlu mengakses database Oracle dan juga database MySQL, Anda harus mendownload kedua modul database Oracle dan MySQL.

API DB menyediakan standar minimal untuk bekerja dengan database menggunakan struktur dan sintaks Python sejauh mungkin. API ini meliputi:

- Mengakses modul API.
- Mendapatkan koneksi dengan database.
- Melakukan pernyataan SQL dan prosedur tersimpan.
- Menutup koneksi

Python memiliki dukungan built-in untuk MySQL. Untuk mengakses MySQL, Anda perlu selesaikan konsep menggunakan MySQL. Modul MySQLdb, antarmuka yang populer dengan MySQL, tidak kompatibel dengan Python 3. Untuk gantinya, kita akan menggunakan modul PyMySQL.

Anda bisa memilih database yang tepat untuk aplikasi Anda. API Database Python mendukung berbagai macam server database seperti.

- GadFly
- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase
- SQLite

Sebagai contoh, jika Anda perlu mengakses database Oracle dan juga database MySQL, Anda harus mendownload kedua modul database Oracle dan MySQL

API DB menyediakan standar minimal untuk bekerja dengan database menggunakan struktur dan sintaks Python sedapat mungkin.
API ini meliputi:

- Mengimpor modul API.
- Mendapatkan koneksi dengan database.
- Menerbitkan pernyataan SQL dan prosedur tersimpan.
- Menutup koneksi

Python memiliki dukungan built-in untuk SQLite. Pada bagian ini, kita akan mempelajari semua konsep menggunakan MySQL. Modul MySQLdb, antarmuka yang populer dengan MySQL tidak kompatibel dengan Python 3. Sebagai gantinya, kita akan menggunakan modul PyMySQL.

APA ITU PYMYSQL ?

PyMySQL adalah sebuah antarmuka untuk menghubungkan ke server database MySQL dari Python. Ini mengimplementasikan API Database Python v2.0 dan berisi perpustakaan klien MySQL murni-Python. Tujuan PyMySQL adalah penggantian drop-in untuk MySQLdb.
Dapat melihat dokumentasi lengkap penggunaan PyMySQL di <https://pymysql.readthedocs.io/en/latest/>.

CARA INSTAL PYMYSQL

Sebelum melanjutka, pastikan Anda telah menginstal PyMySQL di komputer Anda. Cukup ketik berikut ini di skrip Python Anda dan jalankan.

```
import pymysql.cursors
```

Jika menghasilkan hasil berikut, berarti modul MySQLdb tidak terpasang:

```
Traceback (most recent call last): File "test.py", line 3,  
in Import PyMySQL ImportError: No module named  
PyMySQL
```

Untuk menginstal modul PyMySQL silahkan gunakan command/perintah berikut di command prompt:

```
python -m pip install PyMySQL
```

DATABASE CONNECTION

Sebelum terhubung ke database MySQL,
pastikan beberapa hal dibawah ini :

- Anda telah membuat database TESTDB.
- Anda telah membuat tabel EMPLOYEE di TESTDB.
- Tabel ini memiliki bidang FIRST_NAME, LAST_NAME, AGE, SEX, dan INCOME.
- User ID “testuser” dan password “test123” diatur untuk mengakses TESTDB.
- Python modul PyMySQL terinstal dengan benar pada mesin Anda.
- Anda telah melalui tutorial MySQL untuk memahami Dasar-Dasar MySQL

DATABASE CONNECTION II

Berikut ini adalah contoh koneksi dengan database MySQL “TESTDB”

```
import pymysql.cursors

# Open database connection
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.
data = cursor.fetchone()

print ("Database version : %s " % data)

# disconnect from server
db.close()
```

Membuat Tabel Database

```
import pymysql.cursors

# Open database connection
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Drop table if it already exist using execute() method.
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# Create table as per requirement
sql = """CREATE TABLE EMPLOYEE (
      FIRST_NAME  CHAR(20) NOT NULL,
      LAST_NAME  CHAR(20),
      AGE INT,
      SEX CHAR(1),
      INCOME FLOAT )"""

cursor.execute(sql)

# disconnect from server
db.close()
```

Operasi Insert

```
import pymysql.cursors

# Open database connection
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
sql = """INSERT INTO EMPLOYEE(FIRST_NAME,
   LAST_NAME, AGE, SEX, INCOME)
   VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""
try:
    # Execute the SQL command
    cursor.execute(sql)
    # Commit your changes in the database
    db.commit()
except:
    # Rollback in case there is any error
    db.rollback()

# disconnect from server
db.close()
```

Read Operation

```
import pymysql.cursors

# Open database connection
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
sql = "SELECT * FROM EMPLOYEE \
      WHERE INCOME > '%d'" % (1000)
try:
    # Execute the SQL command
    cursor.execute(sql)
    # Fetch all the rows in a list of lists.
    results = cursor.fetchall()
    for row in results:
        fname = row[0]
        lname = row[1]
        age = row[2]
        sex = row[3]
        income = row[4]
        # Now print fetched result
        print ("fname = %s,lname = %s,age = %d,sex = %s,income = %d" % \
              (fname, lname, age, sex, income ))
except:
    print ("Error: unable to fetch data")

# menutup koneksi ke server
db.close()
```

Update Operation

```
import pymysql.cursors

# Open database connection
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to UPDATE required records
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1
      WHERE SEX = '%c'" % ('M')

try:
    # Execute the SQL command
    cursor.execute(sql)
    # Commit your changes in the database
    db.commit()
except:
    # Rollback in case there is any error
    db.rollback()

# disconnect from server
db.close()
```

Delete Operation

```
import pymysql.cursors

# Open database connection
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to DELETE required records
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
    # Execute the SQL command
    cursor.execute(sql)
    # Commit your changes in the database
    db.commit()
except:
    # Rollback in case there is any error
    db.rollback()

# disconnect from server
db.close()
```

13. Pemrograman Berorientasi Objek

