ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ:

1. Python, SQL
2. Python Modules: 1.3 (decimal), 3.1 (datetime)
3. Python Packages: Python+MySQL
4. Python Advanced: 1 (PIP/PyPI)

ΠΕΡΙΕΧΟΜΕΝΑ:

Ευάγγελος Μ.

Σμαραγδένιος Χορηγός Μαθήματος

Τάσος Σαντοριναίος

Σμαραγδένιος Χορηγός Μαθήματος

ΜΑΘΗΜΑ 1: CORE: Ερωτήματα

1. Εισαγωγή

SQLAlchemy psounis

- Η **SQLAlchemy** αποτελείται από 2 κυρίως μέρη:
  - **Core:** Αφαίρεση επί μίας σχεσιακής βάσης. Γράφουμε και εκτελούμε εντολές SQL μέσω της Python
    - Κάθε εντολή SQL έχει δική της σύνταξη στην Python
  - **ORM** (Object Relational Mapper): Απεικόνιση σχεσιακής βάσης σε αντικείμενα.
    - Κατασκευάζονται αντικείμενα τα οποία απεικονίζουν τις αντίστοιχες οντότητες της βάσης δεδομένων (OOP τρόπος για CRUD ενέργειες)

Εγκατάσταση της SQLAlchemy:

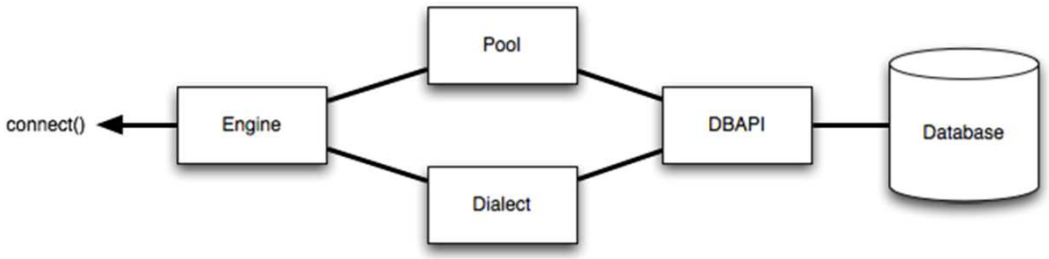
- Απαιτείται το πακέτο σύνδεσης με τη βάση, π.χ. για τη MySQL:

```
> pip install mysql-connector-python
```

- εγκαθιστούμε την SQLAlchemy:

```
> pip install sqlalchemy
```

- Η SQLAlchemy συνεργάζεται με όλα τα κυρίαρχα ΣΔΒΔ
- Απλά απαιτείται η αλλαγή του connector (https://docs.sqlalchemy.org/en/14/core/engines.html)



Παράδειγμα 1: connection.py

dialect driver dbuser dbpassword host database

```
from sqlalchemy import create_engine

conn_str = 'mysql+mysqlconnector://pythonuser:pwd1234@localhost/pyworld'
engine = create_engine(conn_str)
metadata = MetaData()
city = Table('city', metadata, autoload=True, autoload_with=engine)
conn = engine.connect()
```

Ορισμοί:

- **DBAPI (driver):** Σύνδεση με βάση δεδομένων (εδώ ο MySQL Connector)
- **Dialect:** Διαφορετική για κάθε ΣΔΒΔ (χρησιμοποιούμε MySQL)
- **Engine:** Γενική κλάση που εγκαθιστά σύνδεση, τρέχει ερωτήματα, επιστρέφει αποτελέσματα κ.ο.κ.
- **Connection:** Σύνδεση με τη Βάση Δεδομένων (διατηρείται ένα πλήθος συνδέσεων στο Pool, οι οποίες επαναχρησιμοποιούνται)
- **MetaData:** Εσωτερικής χρήσης κλάση της SQLAlchemy στο οποίο απεικονίζουν τις πληροφορίες που κρίνουν απαραίτητες για την περιγραφή της βάσης.
- **Table:** Κλάση που περιτυλίσσει όλες τις πληροφορίες ενός πίνακα (στήλες, τύποι δεδομένων στηλών, περιορισμοί, όχι όμως δεδομένα)
- **connect()** μέθοδος της Engine που εγκαθιστά τη σύνδεση.

Εκτέλεση Ερωτήματος

- Γράφουμε το ερώτημα, με το ιδιότυπο συντακτικό της SQLAlchemy, π.χ. το ακόλουθο επιλέγει τις 5 πρώτες εγγραφές του πίνακα city:

```
query = city.select().limit(5)
```

- Εκτελούμε το ερώτημα με την εντολή:

```
result_set = conn.execute(query)
```

- Παίρνουμε το αποτέλεσμα με μία από τις παρακάτω μεθόδους (επί του αντικειμένου - result_set):

Μέθοδος	Επεξήγηση
fetchone()	Επόμενη εγγραφή (ή None)
fetchmany(n)	Επόμενες n εγγραφές (ή None)
fetchall()	Όλες οι εγγραφές (ή None)
first()	Πρώτη εγγραφή
keys()	ονόματα στηλών που επιστρέφονται
for row in result_set	iterator επί των εγγραφών

  - Κάθε γραμμή του result set είναι ένα tuple τιμών. Έχουμε πρόσβαση στα πεδία:
    - είτε με αρίθμηση, π.χ. row[0], row[1] κ.λπ.
    - είτε ως λεξικό, π.χ. row[“col1”], row[“col2”]
    - είτε με μέλη π.χ. row.col1, row.col2
    - είτε μέσω αντ. Column π.χ. row[table.c.col1]

Παράδειγμα 2: query_run

```
query = city.select().limit(5)
result_set = conn.execute(query)
for row in result_set:
    print(row)
    print(row[0], row[1])
    print(row["ID"], row["Name"])
    print(row.ID, row.Name)
    print(row[city.c.ID], row[city.c.Name])
```

Απλά Ερωτήματα ενός πίνακα

- Κάθε αντικείμενο - πίνακας έχει ως μέλος το columns (και συντομογραφικά c) που με τη σειρά του έχει ως μέλη, τις στήλες του πίνακα (case-sensitive μέλος ή πρόσβαση ως λεξικό)
- Η select (στο sqlalchemy.sql) (εναλλακτική της table.select()) χρησιμοποιείται για να μοντελοποιήσει ερωτήματα ως εξής:

SQL	SQL Alchemy
SELECT * **FROM** city;	select([city])
SELECT ID, Name **FROM** city;	select([city.c.ID, city.c.Name])
SELECT DISTINCT continent **FROM** country;	select([country. c["Continent"]]).distinct()

Παράδειγμα 3: basic_queries (βλ. βίντεο)

Συνώνυμα και Υπολογιζόμενες Στήλες

- Ορίζουμε συνώνυμο σε μία στήλη ως εξής:

SQL	SQL Alchemy
SELECT ID AS city_id, Name FROM city;	select([city.c["ID"].label("city_id"), city.c["Name"]])

- Οι υπολογιζόμενες στήλες, πρέπει να εντίθενται σε παρενθέσεις και να ορίζεται συνώνυμο (αλλιώς τίθεται ένα προκαθορισμένο όνομα)

SQL	SELECT name AS country, 2021-IndepYear AS years_free FROM country;
SQL Alchemy	select(country.c["Name"].label("country"), (2021-country.c["IndepYear"]).label("years_free"))

- συναρτήσεις ΣΔΒΔ (εποπτικά):
  - παρέχεται πρόσβαση στις συναρτήσεις του ΣΔΒΔ μέσω του module func, π.χ. για την concat:

SQL	SELECT CONCAT(HeadOfState, ' of ', name) AS ruler FROM country;
SQL Alchemy	select(func.concat(country.c["HeadOfState"], " of ", country.c["Name"]).label("country"))

Απλές Συνθήκες

- Για να ορίσουμε απλές συνθήκες επεκτείνουμε τη select καλώντας τη μέθοδο where.
- Παραδείγματα με σχεσιακούς τελεστές:

SQL	SELECT productName, quantityInStock FROM products WHERE quantityInStock >= 9000;
SQL Alchemy	select([products.c["productName"], products.c["quantityInStock"]]).\ where(products.c["quantityInStock"] >= 9000)

SQL	SELECT * FROM products WHERE productVendor = 'WELLY DIECAST PRODUCTIONS';
SQL Alchemy	select(products).\ where(products.c["productVendor"] == "WELLY DIECAST PRODUCTIONS")

SQL	SELECT orderNumber, requiredDate, status FROM orders WHERE orderDate <> '2005-05-16';
SQL Alchemy	select([orders.c["orderNumber"], orders.c["orderDate"], orders.c["status"]]).\ where(orders.c["orderDate"] != date(2005, 5, 31))

Παράδειγμα 4: alias_calc_func (βλ. βίντεο)

Παράδειγμα 5: conditions_simple (βλ. βίντεο)

Σύνθεση Συνθηκών με Λογικούς Τελεστές

- **Α' τρόπος:** Διαδοχικές where() ενώνουν τις συνθήκες με AND

SQL
```
SELECT * FROM products
WHERE productLine = 'Classic Cars'
   AND productVendor = 'Min Lin Diecast' ;
```

SQL Alchemy
```
select([products]).\
where(products.c["productLine"] == "Classic Cars").\
where(products.c["productVendor"] == "Min Lin Diecast")
```

- **Β' τρόπος:** Με χρήση τελεστών bit: **& (and), | (or), ~ (and)**

SQL
```
SELECT * FROM products
WHERE productLine = 'Classic Cars'
      OR NOT productVendor = 'Min Lin Diecast';
```

SQL Alchemy
```
select([products]).\
where((products.c["productLine"] == "Classic Cars") \
     | ~ (products.c["productVendor"] == "Min Lin Diecast"))
```

- **Γ' τρόπος:** Με χρήση συναρτήσεων: **and_(), or_(), not()**

SQL
```
SELECT * FROM products
WHERE productLine = 'Classic Cars'
      OR NOT productVendor = 'Min Lin Diecast';
```

SQL Alchemy
```
select([products]).\
where(or_(products.c["productLine"] == "Classic Cars"),
          (not_(products.c["productVendor"]=="Min Lin Diecast")))
```

Παράδειγμα 6: conditions_complex (βλ. βίντεο)

Σύνθετες Συνθήκες

- Όρια τιμών με τη BETWEEN (μέθοδος between()):

SQL
```
SELECT * FROM orderdetails
WHERE priceEach BETWEEN 100 AND 120;
```

SQL Alchemy
```
select([orderDetails]).\
where(orderDetails.c["priceEach"].between(100, 120))
```

- Συγκεκριμένες Τιμές με την IN (μέθοδοι .in_() και .notin_())

SQL
```
SELECT * FROM offices
WHERE city NOT IN ('Boston', 'NYC');
```

SQL Alchemy
```
select([offices]).\
where(offices.c["city"].notin_(["Boston", "NYC"]))
```

- Έλεγχος NULL (προσοχή, όχι με is [not], αλλά με ==, ή !=)

SQL
```
SELECT * FROM offices
WHERE comments IS NOT NULL;
```

SQL Alchemy
```
select([orders]).\
where(orders.c["comments"] != None)
```

- Ταίριασμα συμβολοσειρών: LIKE - REGEXP

SQL
```
SELECT * FROM orders
WHERE comments LIKE ('Cust%')
AND  comments REGEXP ('(the.*){4}');
```

SQL Alchemy
```
select([orders]).\
where(orders.c["comments"].like("Cust%")
          & orders.c["comments"].op("regexp")("(the.*){4}"))
```

Παράδειγμα 7: conditions_complex2 (βλ. βίντεο)

Ταξινόμηση

- Με τη μέθοδο order_by(col):

SQL
```
SELECT * FROM country
WHERE Region = 'Caribbean'
ORDER BY name;
```

SQL Alchemy
```
select([country]).\
where(country.c["Region"] == "Caribbean").\
order_by(country.c["Name"])
```

- Ταξινόμηση σε φθίνουσα σειρά με τη συνάρτηση desc:

SQL
```
SELECT * FROM country
WHERE Region = 'Caribbean'
ORDER BY name DESC;
```

SQL Alchemy
```
select([country]).\
where(country.c["Region"] == "Caribbean").\
order_by(desc(country.c["Name"]))
```

- Ταξινόμηση με βάση (2 ή παραπάνω) στήλες

SQL
```
SELECT Name, Region
FROM country
ORDER BY Region, Name ;
```

SQL Alchemy
```
select([country.c["Name"], country.c["Region"]]).\
order_by(country.c["Region"], desc(country.c["Name"]))
```

Παράδειγμα 8: order_by (βλ. βίντεο)

Όριο Εγγραφών

- Όριο εγγραφών με τη LIMIT

SQL
```
SELECT *
FROM city
ORDER BY ID
LIMIT 100;
```

SQL Alchemy
```
select([city]).\
order_by(city.c["ID"]).\
limit(100)
```

- Όριο εγγραφών με offset αρχής (αρίθμηση εγγραφών από το 0)

SQL
```
SELECT *
FROM city
ORDER BY ID
LIMIT 100, 100;
```

SQL Alchemy
```
select([city]).\
    order_by(city.c["ID"]).\
    limit(100).offset(100)
```

Παράδειγμα 9: limit (βλ. βίντεο)

```
query = select([city]).\
    order_by(city.c["ID"]).\
    limit(100).offset(100)
result_set = conn.execute(query)
for row in result_set:
  print(row)
```

Ομαδοποίηση

- Οι συναρτήσεις σώρευσης βρίσκονται στο module func:

SQL
```
SELECT COUNT(IndepYear)
FROM country;
```

SQL Alchemy
```
select([func.count(country.c["IndepYear"]).label("count")])
```

- Ειδικά για την count(*) πρέπει να χρησιμοποιήσουμε τη select_from(table):

SQL
```
SELECT COUNT(*)
FROM country;
```

SQL Alchemy
```
select([func.count("*").label("count")]).\
select_from(country)
```

- Η GROUP BY ορίζεται ως εξής:

SQL
```
SELECT Continent, Region, COUNT(*) AS Countries
FROM country
GROUP BY Continent, Region;
```

SQL Alchemy
```
select([country.c["Continent"], country.c["Region"],
        func.count("*").label("Countries")]).\
group_by(country.c["Continent"], country.c["Region"])
```

Άσκηση 1:
Μετατρέψτε σε SQLAlchemy το ερώτημα:

```
SELECT Continent, Region, COUNT(*) AS Countries
FROM country
WHERE Continent IN ('Asia', 'Europe', 'Africa')
GROUP BY Continent, Region
ORDER BY Continent, Region DESC
LIMIT 10;
```

Περιορισμοί με τη HAVING

SQL
```
SELECT Continent, COUNT(*) AS Countries
FROM country
GROUP BY Continent
HAVING Countries > 50;
```

SQL Alchemy
```
select([country.c["Continent"],
        func.count("*").label("Countries")]).\
group_by(country.c["Continent"]).\
having(func.count("*") > 50)
```

Παράδειγμα 10: group_by (βλ. βίντεο)
Παράδειγμα 11: having (βλ. βίντεο)

Άσκηση 2:
Μετατρέψτε σε SQLAlchemy το ερώτημα:

```
SELECT Continent, COUNT(*) AS Countries,
       AVG(Population) as avg_population
FROM country
WHERE IndepYear IS NOT NULL
GROUP BY Continent
HAVING Countries > 10
   AND MIN(SurfaceArea) > 10
ORDER BY Countries DESC
LIMIT 2 ;
```