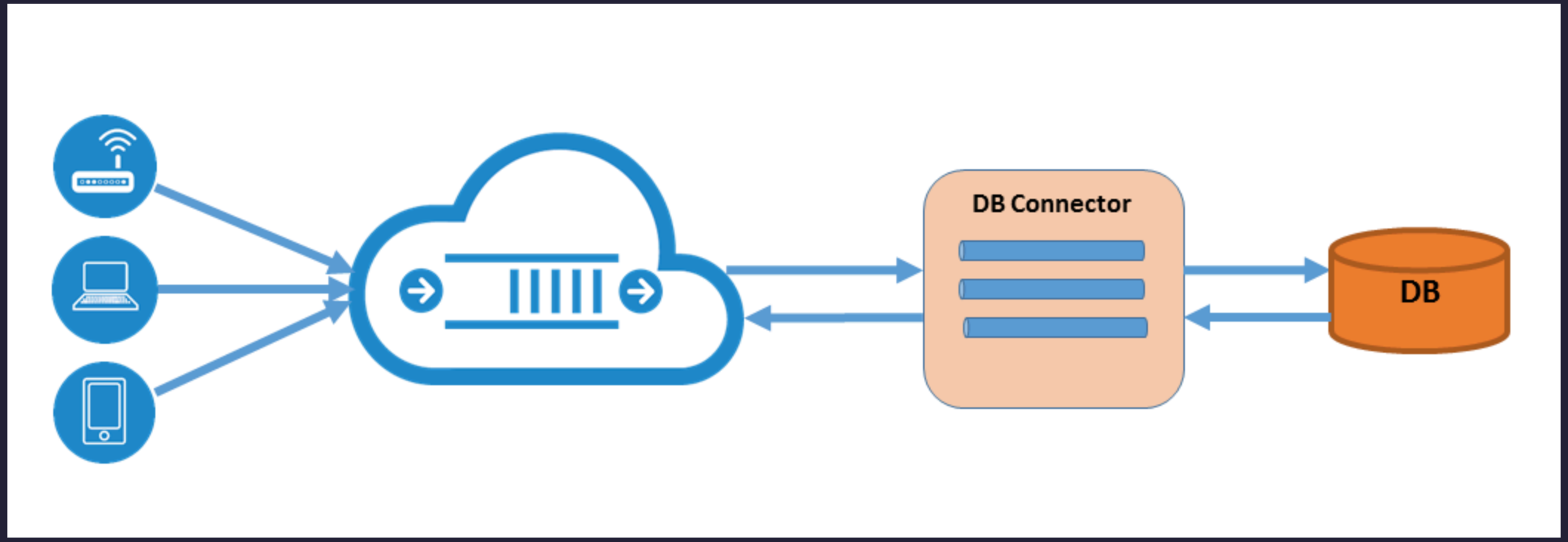
SQL in Python

Translator between Python and SQL



`import` connector package

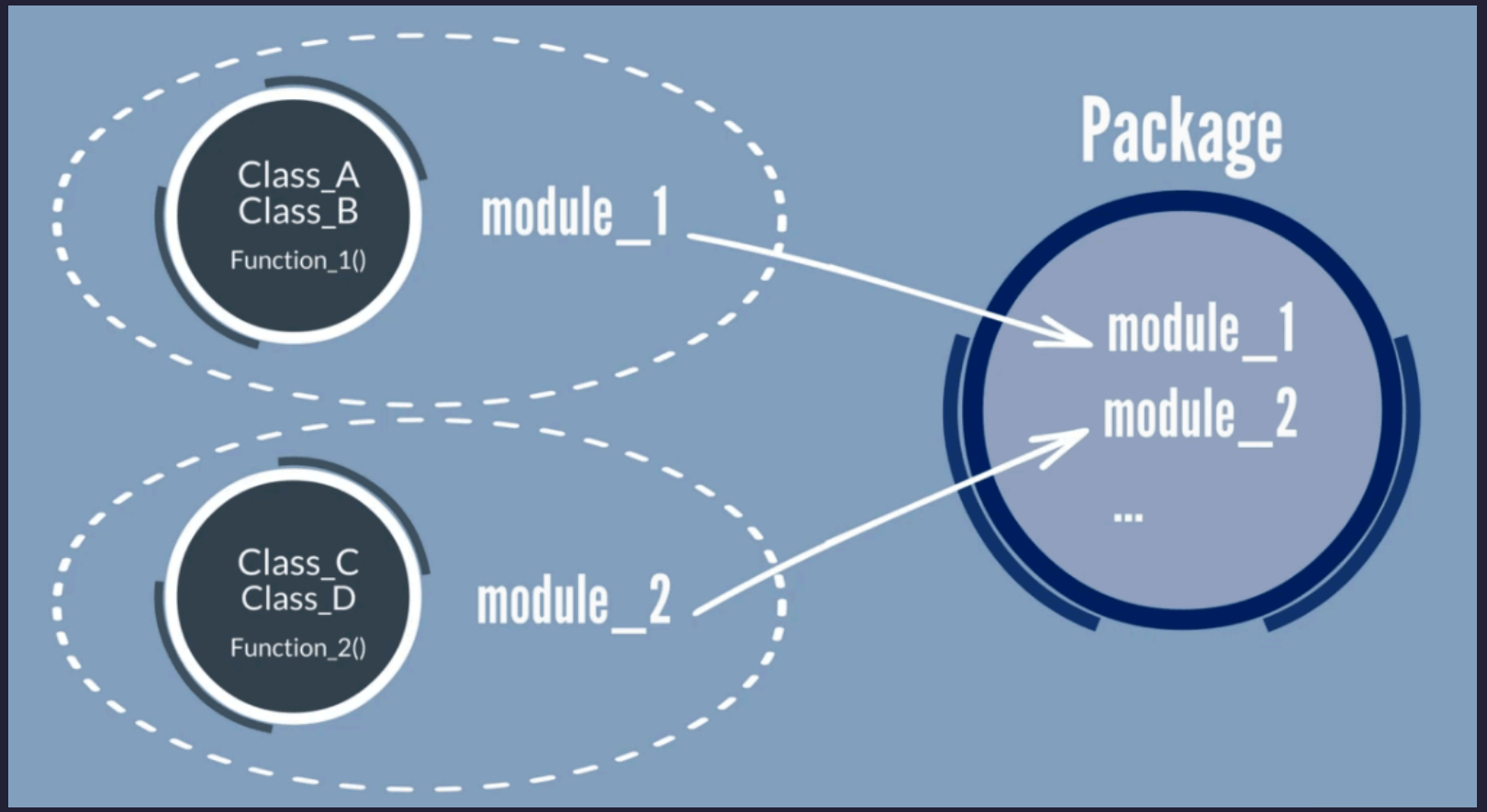
```
import sqlite3

conn = sqlite3.connect('harrypotter.db')

query = "SELECT * FROM students"
result = conn.execute(query).fetchall()

print(result)
```

functions < modules < packages = libraries



Built-in functions: ready to use without `import`

- `print()`
- `type()`
- `len()`
- `range()`
- `input()`

...

---

https://docs.python.org/3/library/functions.html

Built-in modules: no need to install, but need to `import`

- `math`
- `random`
- `datetime`
- `os`
- `sqlite3`

...

---

https://docs.python.org/3/py-modindex.html

`math`

a Python module that provides mathematical functions and constants

https://docs.python.org/3/library/math.html

`import` module

```python
import math

print(math.pi)              # 3.141592653589793
print(math.sqrt(4))         # 2.0
print(math.pow(2, 3))       # 8.0
print(math.floor(3.14))     # 3
print(math.ceil(3.14))      # 4
print(math.factorial(5))    # 120
```

`from` module `import` functions, variables, etc.

```
from math import pi, sqrt

print(pi)
print(sqrt(4))
```

`as` to give alias

```
import math as m

print(m.pi)

print(m.sqrt(4))
```

Install third party packages

- Python Package Index (PyPI; https://pypi.org/)
- `pip install <package_name>`
- `!pip install <package_name>` on Jupyter Notebook

`sqlite3`

a Python package that provides a SQL interface to the SQLite database engine

---

https://docs.python.org/3/library/sqlite3.html

SQLite: Lightweight disk-based database that doesn't require a separate server process

Server process includes:

- access control
- data storage and retrieval
- data backup and recovery
- data dictionary management
- transaction processing (concurrency and recovery)

...

Pros: No installation, no configuration, no maintenance

Cons: Not scalable

Working with databases in Python (DB-API 2.0)

1. Connect to a database (`connect()`)
2. Execute a query (`execute()`)
3. Get query results (`fetchone()`, `fetchmany(n)`, `fetchall()`)
4. Close connection (`close()`)

Connect to a database

`connect()`: create a connection object that enables access to a database

- `connect('name.db')`: load database file `name.db` or create a new one if it doesn't exist

```
import sqlite3

conn = sqlite3.connect('harry-potter.db')
```

Execute a query (DQL)

- `execute()` : execute a query

```
conn.execute("SELECT * FROM students")
```

Get query results

- `fetchone()`: fetch the next row of a query result set, returning a single tuple, or `None` when no more data is available
- `fetchmany(n)`: fetch the next n rows of a query result, returning a list of tuples, or an empty list when no more data is available
- `fetchall()`: fetch all (remaining) rows of a query result, returning a list of tuples

```
one_record = conn.execute("SELECT * FROM students").fetchone()

five_records = conn.execute("SELECT * FROM students").fetchmany(5)

all_records = conn.execute("SELECT * FROM students").fetchall()
```

💻 Query `harrypotter.db` with SQL in Pandas

- connect to `harry-potter.db` using `sqlite3`
- execute queries to answer the following questions:
  - List the name of students who are born after 1980
  - What is the name of the oldest student?
- print query results

`tuple`: like a list, but immutable

`list`: mutable

`tuple`: immutable

```
# list
cities = ["Montreal", "Toronto", "Vancouver", "Detroit"]
print(type(cities))
cities[0] = "New York"

# tuple
cities = ("Montreal", "Toronto", "Vancouver", "Detroit")
print(type(cities))
cities[0] = "New York" # TypeError: 'tuple' object does not support item assignment
```

Execute a query (DDL)

```
query = """
    CREATE TABLE students (
        id INTEGER PRIMARY KEY,
        name TEXT,
        house TEXT,
        age INTEGER
    )
"""
conn.execute(query)
```

Execute a query (DML)

```
query = """
    INSERT INTO students (id, name, house, age)
    VALUES (1, 'Harry Potter', 'Gryffindor', 11)
"""

conn.execute(query)
```

Quotes inside quotes

```
# Error
conn.execute("SELECT * FROM students WHERE first_name = "Harry"")

# Double quotes for outer string
conn.execute("SELECT * FROM students WHERE first_name = 'Harry'")

# Single quotes for outer string
conn.execute('SELECT * FROM students WHERE first_name = "Harry"')

# Escape quotes with escape character (\)
conn.execute("SELECT * FROM students WHERE first_name = \"Harry\"")
```

Commit and close

- `commit()`: commit the current transactions
- `close()`: close the database connection

```
conn.commit()
conn.close()
```

💻 SQL murder mystery

> I don't know her name but I know she's around 5'5" (65") or 5'7" (67"). She has red hair and she drives a Tesla Model S.

1. What's her license id?
2. What's her person id? Use subquery.
3. What's her person id? Use Join.
4. List the people who attended the SQL Symphony Concert in December 2017, along with the number of times each person attended.