# Python SQLite

class Employee:

"""A sample Employee class"""

def \_\_init\_\_(self, first, last, pay):

self.first = first

self.last = last

self.pay = pay

@property

def email(self):

return '{}.{}@email.com'.format(self.first, self.last)

@property

def fullname(self):

return '{} {}'.format(self.first, self.last)

def \_\_repr\_\_(self):

return "Employee('{}', '{}', {})".format(self.first,

self.last, self.pay)

* Cursors facilitate subsequent processing in conjunction with the traversal, such as retrieval, addition and removal of database records
* c.execute("INSERT INTO employees VALUES (?, ?, ?)", (first, last, pay))
* c.execute("INSERT INTO employees

VALUES(:first, :last, :age)", {"first":first, "last":last, "age":pay})

# Create a table

import sqlite3

from employee import Employee

*# To use the module, first create a Connection object that represents the*

*# database. Here the data will be stored in the employee.db file:*

*# You can also supply the special name :memory: to create a database in RAM.*

conn = sqlite3.connect("employees.db") *# establish the connection*

c = conn.cursor()

c.execute(""" CREATE TABLE employees (

                first text,

                last text,

                pay integer

         )""")

conn.commit() *# save the changes*

conn.close() *# close the connection*

# INSERT, SELECT & FETCH records in db

import sqlite3

from employee import Employee

conn = sqlite3.connect("employee.db")

c = conn.cursor()

e1 = Employee("Basir", "johnny", 700000)

e2 = Employee("Bashir", "Payenda", 5000000)

c.execute("INSERT INTO employees VALUES (?, ?, ?)", (e1.first, e1.last, e1.pay))

conn.commit()

c.execute("INSERT INTO employees VALUES(:first, :last, :age)", {"first":e2.first, "last":e2.last, "age":e2.pay})

# commit (namely, save) the changes

conn.commit()

# Note this line didn't work on my machine, seems to be working on others

c.execute("SELECT \* FROM employees WHERE last =?", ('johnny',))

c.execute("SELECT \* FROM employees WHERE last = :last", {"last":"Payenda"})

print(c.fetchall())

# conn.commit()

conn.close()

# 4 methods for CRUD (CREATE, READ, UPDATE, DELETE)

import sqlite3

from employee import Employee

conn = sqlite3.connect(':memory:')

c = conn.cursor()

c.execute("""CREATE TABLE employees (

first text,

last text,

pay integer

)""")

def insert\_emp(emp):

# since we are using context manager, we don’t need **conn.commit()** at the end

with conn:

c.execute("INSERT INTO employees VALUES (:first, :last, :pay)",

{'first': emp.first, 'last': emp.last, 'pay': emp.pay})

def get\_emps\_by\_name(lastname):

c.execute("SELECT \* FROM employees WHERE last=:last",

{'last': lastname})

return c.fetchall()

def update\_pay(emp, pay):

with conn:

c.execute("""UPDATE employees SET pay = :pay

WHERE first = :first AND last = :last""",

{'first': emp.first, 'last': emp.last, 'pay': pay})

def remove\_emp(emp):

with conn:

c.execute("DELETE from employees WHERE first = :first AND

last = :last", {'first': emp.first, 'last': emp.last})

emp\_1 = Employee('John', 'Doe', 80000)

emp\_2 = Employee('Jane', 'Doe', 90000)

insert\_emp(emp\_1)

insert\_emp(emp\_2)

emps = get\_emps\_by\_name('Doe')

print(emps)

update\_pay(emp\_2, 95000)

remove\_emp(emp\_1)

emps = get\_emps\_by\_name('Doe')

print(emps)

conn.close()