OCTOBER 27 – NOVEMBER 2, 2019

المبادرة السعودية للمطورين

مسار الـPython

مشرفي المسار:

عبدالله عوده – انتصار النصار – رؤى كردي – لينا المصعبي

ملاحظات قبل بدء الدروس:

- على المتدربين نشر كل يوم الجزئية التي تم كتابتها من النص البرمجي في الـ Github تحت Topic بعنوان saudidevorg

كما تم توضيحه في دروس الـ Github سابقاً

- على المتدربين نشر كل يوم مقدار التقدم وصورة لما تم تعلمه وتطبيقه على Twitter تحت الهاشتاقات:

#المبادرة_السعودية_للمطورين

#100_يوم_برمجة

#100DaysOfCode

تمنياتنا لك بالتوفيق

المبادرة السعودية للمطورين

اليوم التاسع والستون

التعامل مع الملفات في لغة البايثون

File Handling

فتح الملف في بايثون

➢ Python File Open

File handling is an important part of any web application.

Python has several functions for creating, reading, updating, and deleting files.

التعامل مع الملفات هو جزء مهم في أي تطبيق ويب

وبايثون لديه عدة دوال لإنشاء وقراءة وتحديث وحذف الملفات

التعامل مع الملفات

➢ File Handling

The key function for working with files in **Python** is the **open()** function.

The **open()** function takes two parameters; **filename**, and **mode**.

الدالة الأساسية للتعامل والعمل مع الملفات في بايثون هي دالة **open()**

تعتبر من الدوال الجاهزة في بايثون وتستخدم لإنشاء أو فتح الملف

الدالة تأخذ ٢ من المُعامِلات، اسم الملف وطريقة التعامل مع الملف هل للقراءة أو الكتابة .. إلخ

There are **four** different methods (modes) for opening a file:

هناك أربعة طرق لفتح الملف (الهدف من فتح الملف)

1/ "**r**" - **Read** - Default value. Opens a file for reading, error if the file does not exist.

الرمز **r** هو اختصار لكلمة **Read** فتح الملف لأجل القراءة منه، وهو القيمة الافتراضية للملف الذي تقوم بفتحه

يظهر خطأ إذا لم يكن الملف موجودا

2/ "**a**" - **Append** - Opens a file for appending, creates the file if it does not exist.

الرمز **a** هو اختصار لكلمة **Append** فتح الملف لأجل إضافة نص جديد على النص الموجود سابقا

وإذا لم يكن الملف موجودا سيتم إنشاؤه

3/ "**w**" - **Write** - Opens a file for writing, creates the file if it does not exist.

الرمز **w** هو اختصار لكلمة **Write** فتح الملف لأجل الكتابة، وإذا لم يكن الملف موجودا سيتم إنشاؤه

4/ "**x**" - **Create** - Creates the specified file, returns an error if the file exists.

الرمز **x** هو اختصار لكلمة **Create** فتح الملف لأجل إنشاء ملف جديد، يقوم بترجيع خطأ في حال كان الملف موجودا

In addition, you can specify if the file should be handled as **binary** or **text** mode

"t" - **Text** - Default value. Text mode "b" - **Binary** - Binary mode (e.g. images).

يمكنك تحديد ما إذا كان الملف يُتعامل معه كنص عادي **text** (ملف نصي) وذلك باستخدام الرمز t

أو عبارة عن أي ملفات غير نصية وتسمى **binary** باستخدام الرمز b مثل الصور والفيديوهات ..إلخ

➢ Syntax

طريقة البناء

To open a file for reading it is enough to specify the name of the file.

لفتح الملف للقراءة، يكفي تحديد اسم الملف المراد قراءته

```
f = open("demofile.txt")
```

في الدالة **open()** مكان الباراميتر **filename** قمنا بتمرير نص يمثل اسم الملف الذي نريد إنشاؤه والتعامل معه

The code above is the same as.

الكود البرمجي السابق أعلاه هو نفسه هذا الكود

```
f = open("demofile.txt", "rt")
```

filename **mode**

Because "r" for **read**, and "t" for **text** are the default values, you do not need to specify them.

الرمز r يرمز إلى القراءة، والرمز t يرمز إلى أن قيمة الملف الافتراضية هي نص (ملف نصي)

Note: Make sure the file exists, or else you will get an error.

تأكد بأن الملف موجود أصلا، وإلا سيظهر لك خطأ

➢ Python Read Files

قراءة النص الموجود في ملف نصي في بايثون

➢ Open a File on the Server

فتح الملف على الخادم

Assume we have the following file, located in the same folder as **Python**.

افترض أنه لديك هذا الملف النصي التالي، موقعه في نفس مجلد بايثون

demofile.txt ← أنشأنا ملف نصي و قمنا بتسميته ويكون امتداده .txt

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

← النص الموجود بداخل الملف

To open the file, use the built-in **open()** function.The **open()** function returns a file object, which has a **read()** method for reading the content of the file.

لفتح الملف استخدم الدالة **open()** تقوم بترجيع الكائن الذي يشير للملف والذي يحتوي على الدالة **read()** لقراءة محتوى الملف

Example

```
f = open("demofile.txt", "r")

print(f.read())
```

تشغيل النتيجة في موجه الأوامر

```
C:\Users\My Name>python demo_file_open.py
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

← اسم الملف الذي يحتوي على الأكواد ويكون امتداده .py

← نتيجة التشغيل، سيتم قراءة محتوى الملف demofile.txt

```
Python 3.7.4 (v3.7.4:e09359112e, Ju
[Clang 6.0 (clang-600.0.57)] on dar
Type "help", "copyright", "credits"
>>>
============== RESTART: /Users/Rae
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

*demo_file_open.py - /User

```
f = open("demofile.txt", "r")

print(f.read())
```

← نتيجة تشغيل الكود

➢ Read Only Parts of the File

قراءة جزء من الملف

By default the **read()** method returns the whole text, but you can also specify how many characters you want to return.

الدالة **read()** تقوم بترجيع كامل محتويات الملف افتراضيا، لكن يمكنك تحديد أجزاء محددة من الملف للقراءة

وذلك بتحديد عدد الأحرف التي تريد قراءتها

Example

Return the 5 first characters of the file.

هنا في هذا المثال

```
f = open("demofile.txt", "r")

print(f.read(5))
```

نريد قراءة فقط أول خمسة أحرف من الملف

عبارة عن الكلمة الأولى في الملف النصي demofile.txtt

```
C:\Users\My Name>python demo_file_open2.py
Hello
```

اسم ملف البايثون الذي يحتوي على الكود

نتيجة تشغيل الكود

➢ Read Lines

قراءة الأسطر

You can return one line by using the **readline()** method.

يمكنك قراءة سطر واحد فقط من الملف باستخدام الدالة **readline()**

Example

Read one line of the file.

في هذا المثال

قمنا بقراءة سطر واحد فقط في الملف النصي demofile.txtt

```
f = open("demofile.txt", "r")

print(f.readline())
```

```
C:\Users\My Name>python demo_file_readline.py
Hello! Welcome to demofile.txt
```

نتيجة التشغيل

By calling **readline()** two times, you can read the two first lines.

عند استدعائك للدالة لأكثر من مرة، مرتين مثلًا ستتم قراءة أول سطرين من الملف

Example

Read two lines of the file.

قراءة أول سطرين من الملف

```
f = open("demofile.txt", "r")

print(f.readline())
print(f.readline())
```

```
C:\Users\My Name>python demo_file_readline2.py
Hello! Welcome to demofile.txt
This file is for testing purposes.
```

نتيجة التشغيل

By looping through the lines of the file, you can read the whole file, line by line.

يمكنك عرض جميع الأسطر باستخدام حلقة التكرار **for** وبالتالي يمكنك قراءة كامل الملف سطر بسطر

Example

Loop through the file line by line.

```
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

```
C:\Users\My Name>python demo_file_readline3.py
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

نتيجة التشغيل

➢ Close Files

إغلاق الملفات

It is a good practice to always close the file when you are done with it.

من الممارسات الجيدة هي أنك تقوم بإغلاق الملف بعد الانتهاء منه

Example

في هذا المثال

Close the file when you are finish with it.

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

بعد الانتهاء من قراءة الملف

قمنا باستدعاء الدالة Close() من الكائن f

وذلك لإغلاق الاتصال مع الملف المفتوح في الذاكرة

```
C:\Users\My Name>python demo_file_close.py
Hello! Welcome to demofile.txt
```

نتيجة التشغيل

Note: You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

يجلب عليك دائما إغلاق الملف بعد الانتهاء منه، وتنظيف الذاكرة من كل ماله علاقة بالملف

بحيث أنك لو أردت أن تضيف بعض التغييرات على الملف لن تحدث هذه التغييرات ويتم حفظها إلا بعد إغلاقك للملف

➢ Python File Write

الكتابة في ملف في بايثون

➢ Write to an Existing File

الكتابة في ملف موجود مسبقا

To write to an existing file, you must add a parameter to the **open()** function:

للكتابة على ملف موجود مسبقا، يجب عليك تمرير معاملات لدالة فتح الملف

"a" - **Append** - will append to the end of the file

الرمز a هو اختصار لكلمة Append ستتم الكتابة في آخر الملف

"w" - **Write** - will overwrite any existing content

الرمز w هو اختصار لكلمة Write يقوم بالكتابة وحذف النص الموجود داخل الملف في حال لم يكن الملف فارغا

Example

في هذا المثال

Open the file "demofile2.txt" and append content to the file.

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

قمنا بفتح الملف النصي demofile2.txt

ثم قمنا بكتابة وإضافة نص جديد في آخر الملف

على النص الموجود مسبقا في الملف

وإغلاق الملف بعد الانتهاء من الكتابة عليه

```
C:\Users\My Name>python demo_file_append.py
Hello! Welcome to demofile2.txt
This file is for testing purposes.
Good Luck!Now the file has more content!
```

نتيجة التشغيل

الكتابة الجديدة على الملف

Example

أما هنا في هذا المثال

Open the file "demofile3.txt" and overwrite the content.

قمنا بفتح الملف النصي demofile3.txt

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the writing
#open and read the file after the appending:
f = open("demofile3.txt", "r")
print(f.read())
```

ثم قمنا بكتابة نص جديد في الملف

سيتم حذف النص الموجود مسبقا واستبداله بالنص الجديد الذي تم كتابته في الملف مؤخرا

و من ثم إغلاق الملف بعد الانتهاء من الكتابة

```
C:\Users\My Name>python demo_file_write.py
Woops! I have deleted the content!
```

نتيجة التشغيل

Note: the "w" method will overwrite the entire file.

دالة الكتابة تقوم بالكتابة على الملف كاملا، وحذف النص السابق

➢ Create a New File

إنشاء ملف جديد

To create a new file in **Python**, use the **open()** method, with one of the following parameters:

لإنشاء ملف جديد في بايثون، استخدم دالة الفتح مع أحد هذه المُعامِلات (الباراميتر)

"x" - **Create** - will create a file, returns an error if the file exist

الرمز x يرمز للكلمة Create يقوم بإنشاء ملف، فإذا كان هناك ملف موجود مسبقا يرجّع لك خطأ

"a" - **Append** - will create a file if the specified file does not exist

الرمز a يرمز للكلمة Append يقوم بإنشاء ملف، إذا كان الملف المطلوب/المحدد غير موجود

"w" - **Write** - will create a file if the specified file does not exist

الرمز w يرمز للكلمة Write يقوم بإنشاء ملف، إذا كان الملف المطلوب/المحدد غير موجود

Example

Create a file called "myfile.txt".

في هذا المثال

قمنا بإنشاء ملف نصي باسم myfile.txt

```
f = open("myfile.txt", "x")
```

Result: a new empty file is created!

النتيجة: سيتم إنشاء ملف جديد فارغ

Example

Create a new file if it does not exist.

في هذا المثال

إذا كان الملف myfile.txt موجود مسبقا، فلن يتم إنشاء ملف جديد

```
f = open("myfile.txt", "w")
```

➢ Delete a File

حذف الملف

To delete a file, you must import the **OS** module, and run its **os.remove()** function

لحذف ملف يجب عليك استخدام الوحدة الجاهزة في بايثون OS ، والتي تحتوي على دالة الحذف **remove()**

Example

انظر المثال التالي

Remove the file "demofile.txt".

قمنا بتضمين الوحدة os لاستخدام الدوال الموجودة بداخله

ثم قمنا باستدعاء الدالة **remove()** لمسح الملف demofile.txt

من خلال تمرير اسم الملف المراد حذفه

```
import os
os.remove("demofile.txt")
```

➢ Check if File exist

التحقق ما إذا كان الملف موجودا أم لا

To avoid getting an error, you might want to check if the file exists before you try to delete it.

لتجنب حدوث خطأ، الأفضل أن تقوم بالتحقق ما إذا الملف موجود أم لا؟ وذلك قبل القيام بعملية الحذف

Example

انظر المثال التالي

Check if file exists, then delete it.

قمنا بتضمين الوحدة os لاستخدام الدوال الموجودة فيها

ثم قمنا باستدعاء الدالة **path.exists()**

لمعرفة هل الملف demofile.txt موجودا أم لا؟

فإذا كان موجود نستدعي دالة الحذف **remove()**

لمسح الملف demofile.txt

وإذا لم يكن موجودا سيتم طباعة جملة نصية تفيد بذلك

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

➢ Delete Folder

حذف المجلد

To delete an entire folder, use the **os.rmdir()** method.

Example

لمسح المجلد كاملا، استخدم الدالة **rmdir()** الموجودة في الوحدة OS

Remove the folder "myfolder".

في المثال التالي قمنا بتضمين الوحدة os لاستخدام الدوال الموجودة فيها

ثم قمنا باستدعاء الدالة **rmdir()** لمسح المجلد myfolder

عند تشغيل الكود سيتم مسح المجلد

```
import os
os.rmdir("myfolder")
```

Note: You can only remove ***empty*** folders.

يمكنك حذف المجلد إذا كان فارغا فقط، لا يحتوي على أي ملف بداخله

أتممت درسك بنجاح!

روابط قد تهمك

Useful links

- 7.2. Reading and Writing Files
- Text Files in Python || Python Tutorial || Learn Python Programming
- #65 Python Tutorial for Beginners | File handling
- Reading Files | Python | Tutorial 28
- Writing to Files | Python | Tutorial 29
- تعلم البرمجة بلغة بايثون - Files 19 - الملفات
- Learn Python in Arabic #4 - 4 شرح بايثون بالعربي Python in Arabic حفظ و فتح ملف Save And Open File
- Learn Python in Arabic #47 - 47 شرح بايثون بالعربي Python in Arabic تجهيز الملف file w r open Python
- Learn Python in Arabic #49 - القراءة من الملف read readlines file r open Python
- Learn Python in Arabic #48 - الكتابة علي الملف write writelines file w open Python
- Learn Python in Arabic #50 - كتابة علي ملف موجود append file a open Python
- Learn Python in Arabic #51 - كتابة او قراءة ملف ثنائي read write binary file rb wb open Py
- Learn Python in Arabic #52 - انماط تعامل الملفات modes file open r w a all with b Python
- Learn Python in Arabic #54 - انشاء المجلدات create directory or folder Python
- Learn Python in Arabic #55 - التحقق من وجود ملفات و مجلدات if exists file folder Python
- Learn Python in Arabic #56 - حذف الملفات و المجلدات delete files folders Python
- تعلم بايثون 3 - كتابة وقراءة الملفات #14
- Text Files in Python || Python Tutorial || Learn Python Programming

طبّق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم السبعون

قواعد البيانات في بايثون

Python MySQL

Python can be used in database applications. One of the most popular databases is **MySQL**.

يمكنك استخدام بايثون في تطبيقات قواعد البيانات ومن أشهر قواعد البيانات هي MySQL

➢ MySQL Database

قاعدة البيانات MySQL

To be able to experiment with the code examples in this tutorial, you should have **MySQL** installed on your computer.

You can download a free **MySQL** database at **https://www.mysql.com/downloads/**

في هذا الدرس ستتعلم كيف تتعامل مع قواعد البيانات MySQL

لذلك يلزمك تنصيب قواعد البيانات MySQL من الموقع الرسمي مسبقا في حال لم تكن على جهازك، لتستطيع تطبيق الدروس والأمثلة

1/ Install MySQL server **أولا عليك تحميل قواعد البيانات**

2/ Install MySQL Workbench **ثم قم بتحميل هذا البرنامج للواجهة**

وفي حال كنت لا تملك أي فكرة عن قواعد البيانات قم بمشاهدة دروس لتعلم أساسيات التعامل مع قواعد البيانات

هنا ستجد دورة للمبتدئين في قناة الأستاذ عيد عبدالله MYSQL 101

➢ Install MySQL Driver

تثبيت مشغل قاعدة البيانات MySQL

Python needs a **MySQL driver** to access the **MySQL** database. In this tutorial we will use the driver "**MySQL Connector**".

بايثون تحتاج إلى المشغل MySQL driver للوصول إلى قاعدة البيانات MySQL والتعامل معها

وللتعامل مع قواعد البيانات يجب تنصيب وتثبيت الحزمة MySQL Connector في بايثون

We recommend that you use **PIP** to install "**MySQL Connector**". **PIP** is most likely already installed in your **Python** environment.

ننصحك باستخدام الأداة PIP التي تعرفت عليها في الدروس السابقة لتنزيل وتثبيت الحزمة MySQL Connector

Navigate your command line to the location of **PIP**, and type the following: Download and install "**MySQL Connector**"

في موجه الأوامر الخاص بجهازك اذهب إلى المسار الصحيح للـ PIP في بايثون

ثم اكتب الأمر `pip install mysql-connector` **قد تختلف بعض الأوامر لإختلاف نوع نظام التشغيل للجهاز وإصدارات البايثون**

بعد تثبيتك للحزمة ستكون قادرا على تضمينها واستخدامها

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install mysql-connector
```

Now you have downloaded and installed a **MySQL driver**.

➢ Test MySQL Connector

التأكد من وجود حزمة الاتصال لقواعد البيانات

To test if the installation was successful, or if you already have "**MySQL Connector**" installed, create a **Python** page with the following content.

لاختبار والتأكد من نجاح التنزيل للحزمة MySQL Connector أنشأ صفحة بايثون تحتوي على التالي :

اسم الملف demo_mysql_test.py

```
import mysql.connector
#if this page is executed with no errors, you have the "mysql.connector" module installed.
```

اكتب هذا الكود (النص البرمجي) في ملف بايثون

```
C:\Users\My Name>python demo_mysql_test.py
```

اكتب في موجه الأوامر `python` **أو** `python3` **ثم اسم الملف** `demo_mysql_test.py`

If the above code was executed with no errors,

"**MySQL Connector**" is installed and ready to be used.

إذا تم تنفيذ الكود السابق بدون ظهور أي خطأ

فهذا يعني أن موصل قاعدة البيانات MySQL Connector تم تثبيته و جاهز للاستخدام

➢ Create Connection

إنشاء الاتصال

Start by creating a connection to the database. Use the username and password from your **MySQL** database.

ابدأ بإنشاء اتصال بقاعدة البيانات، استخدم اسم المستخدم وكلمة السر من قاعدة البيانات التي بجهازك

اكتب هذا الكود (النص البرمجي) في ملف بايثون

demo_mysql_connection.py اسم الملف

قمنا بتضمين الحزمة التي تم تثبيتها mysql- connector

ثم استدعينا الدالة connect() منها

لإنشاء كائن يسمح بالاتصال بخادم قواعد البيانات MySQL

الكائن

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword"
)

print(mydb)
```

قاعدة البيانات موجودة على جهازك

اسم المستخدم الذي تم اعتماده أثناء تثبيت قاعدة البيانات

كلمة المرور للمستخدم

الدالة connect() تُستخدم لإنشاء كائن يسمح بالاتصال بقاعدة البيانات MySQL

تحتوي على المُعاملات (الباراميتر) التالية:

host تمرر له المكان الموجود عليه الخادم

user تمرر له اسم المستخدم الذي سيتم من خلاله التعامل مع قاعدة البيانات

passwd تمرر كلمة مرور المستخدم

تستخدم هذه المعلومات دائما لإنشاء الاتصال

قمنا بطباعة هذه المعلومات للتأكد من أن عملية الاتصال بخادم البيانات تمت بنجاح

سيظهر لك خطأ إن لم تكن عملية الاتصال صحيحة

```
C:\Users\My Name>python demo_mysql_connection.py
<mysql.connector.connection.MySQLConnection object ar 0x016645F0>
```

Now you can start querying the database using **SQL** statements.

الآن يمكنك إرسال استعلامات لخادم قاعدة البيانات باستخدام عبارات الـ SQL

➢ Python MySQL Create Database

إنشاء قاعدة بيانات في بايثون

➢ Creating a Database

إنشاء قاعدة بيانات

To create a database in **MySQL**, use the "**CREATE DATABASE**" statement.

لإنشاء قاعدة بيانات في MySQL استخدم عبارة الإنشاء CREATE DATABASE في لغة SQL

Example

create a database named "**mydatabase**".

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE mydatabase")

#If this page is executed with no error, you have successfully created a database.
```

أنشأنا قاعدة بيانات باسم mydatabase

الدالة cursor() تُستدعى من الكائن الذي ترجعه الدالة connect()

لإرسال استعلامات إلى خادم قاعدة البيانات

والحصول على نتيجة الإرجاع التي يرجعها خادم قاعدة البيانات بعد الانتهاء من تنفيذ الاستعلامات

الدالة execute() تُستدعى من الكائن الذي ترجعه الدالة cursor ()

لإرسال استعلامات إلى خادم قاعدة البيانات، نقوم بتمرير الاستعلام عن طريق أقواس الدالة

```
C:\Users\My Name>python demo_mysql_create_db.py
```

اسم ملف بايثون الذي يحتوي على الكود

If the above code was executed with no errors, you have successfully created a database.

إذا لم يظهر لك خطأ بعد تشغيل الملف هذا يعني أنه تم إنشاء قاعدة البيانات بنجاح

➢ Check if Database Exists

التأكد من وجود قاعدة البيانات

You can check if a database exists by listing all databases in your system by using the "**SHOW DATABASES**" statement.

يمكنك التأكد ما إذا كانت قاعدة البيانات موجودة أم لا

وذلك بعرض جميع قواعد البيانات الموجودة

باستخدام عبارة الاستعلام **SHOW DATABASES**

Example

انظر المثال التالي

Return a list of your system's databases.

قمنا بعرض جميع قواعد البيانات الموجودة

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")

for x in mycursor:
  print(x)
```

اسم ملف بايثون الذي يحتوي على الكود

```
C:\Users\My Name>python demo_mysql_show_databases.py
('information_scheme',)
('mydatabase',)
('performance_schema',)
('sys',)
```

نتيجة التشغيل

عرض جميع قواعد البيانات الموجودة

Or you can try to access the database when making the connection.

يمكنك المحاولة للوصول لقاعدة البيانات أثناء عملية إنشاء الاتصال

Example

Try connecting to the database "mydatabase".

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

#If this page is executed with no error, the database "mydatabase" exists in your system
```

مثال لاسم المستخدم نضع الاسم : root

مثال لكلمة مرور المستخدم نكتب : pass1233

اسم قاعدة البيانات التي نريد التعامل معها

```
C:\Users\My Name>python demo_mysql_db_exist.py
```

If the database does not exist, you will get an error.

إذا لم تكن قاعدة البيانات موجودة، سيظهر لك خطأ

➢ Python MySQL Create Table

إنشاء جدول في قاعدة بيانات في بايثون

➢ Creating a Table

إنشاء جدول

To create a table in **MySQL**, use the "**CREATE TABLE**" statement.

Make sure you define the name of the database when you create the connection.

لإنشاء جدول في قاعدة البيانات MySQL استخدم عبارة الإنشاء في لغة SQL "CREATE TABLE"

تأكد أنك قمت بتعريف اسم قاعدة البيانات عند إنشاء الاتصال

Example

في هذا المثال قمنا بإنشاء الجدول customers داخل قاعدة البيانات mydatabase

Create a table named "**customers**".

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")

#If this page is executed with no error, you have successfully created a table named "customers".
```

استدعاء الدالة connect() للاتصال بقاعدة البيانات

استدعاء الدالة cursor () للتعامل مع قاعدة البيانات

```
C:\Users\My Name>python demo_mysql_create_table.py
```

If the above code was executed with no errors, you have now successfully created a table.

إذا تم تنفيذ الكود السابق بدون أي خطأ، فهذا يعني أنك قمت بإنشاء الجدول في قاعدة البيانات بنجاح

➢ Check if Table Exists

التأكد من وجود الجدول

You can check if a table exist by listing all tables in your database with the "**SHOW TABLES**" statement.

يمكنك التأكد ما إذا كان الجدول موجودا في قاعدة البيانات أم لا

وذلك بعرض جميع الجداول الموجودة في قواعد البيانات

باستخدام عبارة الاستعلام **SHOW TABLES**

Example

Return a list of your system's databases tables.

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

for x in mycursor:
     print(x)
```

```
C:\Users\My Name>python demo_mysql_show_tables.py
('customers',)
```

نتيجة التشغيل

جميع الجداول الموجودة

➢ Primary Key المفتاح الأساسي

When creating a table, you should also create a column with a **unique key** for each record.

This can be done by defining a **PRIMARY KEY**.

عند إنشائك للجدول، يجب عليك أيضا إنشاء عامود مع مفتاح فريد لكل صف، يتم تنفيذ ذلك باستخدام المفتاح الأساسي

We use the statement "**INT AUTO_INCREMENT PRIMARY KEY**" which will insert a unique number for each record. Starting at **1** and increased by one for each record.

نستخدم الأمر أو العبارة **INT AUTO_INCREMENT PRIMARY KEY**

لإضافة عدد فريد لكل صف، بداية من العدد واحد 1 ويزيد في كل مرة بواحد لكل صف

Example

Create **primary key** when creating the table.

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255))")

#If this page is executed with no error, the table "customers" now has a primary key
```

إذا تم تنفيذ الكود السابق بدون أي خطأ، فهذا يعني بأن الجدول customers أصبح يحتوي على المفتاح الأساسي

```
C:\Users\My Name>python demo_mysql_primary_key.py
```

نتيجة التشغيل، لا يوجد أخطاء

If the table already exists, use the **ALTER TABLE** keyword.

إذا كان الجدول موجود مسبقا في الأصل استخدم الأمر أو العبارة **ALTER TABLE**

Example

في هذا المثال

Create **primary key** on an existing table.

أنشأنا مفتاح أساسي للجدول الموجود مسبقا customers

في قاعدة البيانات mydatabase

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY")

#If this page is executed with no error, the table "customers" now has a primary key
```

```
C:\Users\My Name>python demo_mysql_alter_primary_key.py
```

نتيجة التشغيل، لا يوجد أخطاء

أتممتَ درسك بنجاح!

تابع التقدّم

روابط قد تساعدك

Check the links below

- **Python and MySQL - Getting Started with MySQL**
- **#72 MySQL Workbench Setup | Python Database Connection**
- **#73 Python Database Connection | MySQL**
- **Python and MySQL - Creating our Database and Table**
- **227 تمهيد قواعد البيانات مع بايثون Database in Python**
- **237 انشاء قاعدة بيانات ب بايثون create database using python**
- **239 عرض جميع قواعد البيانات ب بايثون show all mysql databases using python**
- **240 انشاء جدول ل قاعدة البيانات ب بايثون create table mysql using python**
- **قواعد البيانات Python DataBases 110**

طبّق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم الواحد والسبعون

قواعد البيانات في بايثون ٢

Python MySQL 2

➢ Python MySQL Insert Into Table

الإضافة للجدول في قاعدة بيانات في بايثون

➢ Insert Into Table

الإضافة للجدول

To fill a table in **MySQL**, use the "**INSERT INTO**" statement.

للتعبئة والإضافة في الجدول في قواعد البيانات استخدم الاستعلام INSERT INTO

Example

Insert a record in the "customers" table.

هنا في المثال

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")

mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")
```

قمنا بإضافة سطر/سجل جديد (صف) في الجدول customers الذي أنشأناه في أمثلة الدرس السابق

هنا كتبنا الاستعلام الذي يمكّننا من إضافة سطر/سجل جديد في الجدول customers

val عبارة عن صف tuple سيتم وضع القيم التي فيه داخل الجدول بالترتيب

قمنا باستدعاء الدالة execute() وتمرير الاستعلام المخزن في sql والقيم التي سوف يتم دمجها مع الاستعلام والمخزنة في val

هذه الدالة تستخدم لحفظ المتغيرات التي تمت في قاعدة البيانات

قمنا بطباعة عدد الأسطر التي تمت إضافتها للجدول

إذا لم يظهر خطأ بعد تشغيلك للملف، فهذا يعني أنه تم إضافة سطر جديد في الجدول customers
وسيتم طباعة الجملة التي تعطينا عدد الأسطر المضافة

```
C:\Users\My Name>python demo_mysql_insert.py
1 record inserted.
```

نتيجة التشغيل

Important!: Notice the statement: **mydb.commit()**. It is required to make the changes, otherwise no changes are made to the table.

يجب عليك استدعاء الدالة commit() لحفظ التغيرات التي تقوم بها من إضافة أو تعديل أو مسح من الجدول في قاعدة البيانات

➢ Insert Multiple Rows

إضافة عدة أسطر/صفوف

To insert multiple rows into a table, use the **executemany()** method. The second parameter of the **executemany()** method is a list of tuples, containing the data you want to insert.

لإضافة عدة صفوف في الجدول استخدم الدالة **executemany()** المُعامل (البراميتر) الثاني من هذه الدالة هو عبارة عن قائمة من الصفوف تحتوي على القيم والبيانات التي نريد إضافتها، بمعنى أنه سيتم إنشاء قائمة list وكل عنصر في هذه القائمة هو عبارة عن صف tuple

Example

Fill the "customers" table with data.

هنا في المثال

قمنا بتعبئة الجدول customers بالبيانات

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = [
 ('Peter', 'Lowstreet 4'),
 ('Amy', 'Apple st 652'),
 ('Hannah', 'Mountain 21'),
 ('Michael', 'Valley 345'),
 ('Sandy', 'Ocean blvd 2'),
 ('Betty', 'Green Grass 1'),
 ('Richard', 'Sky st 331'),
 ('Susan', 'One way 98'),
 ('Vicky', 'Yellow Garden 2'),
 ('Ben', 'Park Lane 38'),
 ('William', 'Central st 954'),
 ('Chuck', 'Main Road 989'),
 ('Viola', 'Sideway 1633')
]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "record was inserted.")
```

القيم التي سيتم وضعها داخل الجدول بالترتيب

إذا لم يظهر خطأ بعد تشغيلك للملف، فهذا يعني أنه تم إضافة ١٣ سطر جديد في الجدول customers
وسيتم طباعة الجملة التي تعطينا عدد الأسطر المضافة

```
C:\Users\My Name>python demo_mysql_insert_many.py
13 record was inserted.
```

نتيجة التشغيل

➢ Get Inserted ID

إضافة رقم التعريف

You can get the **id** of the row you just inserted by asking the cursor object.

Note: If you insert more than one row, the **id** of the last inserted row is returned.

يمكنك الحصول على رقم التعريف للصف/سطر الذي قمت بإضافته وذلك عن طريق الكائن cursor

لاحظ: إذا قمت بإضافة أكثر من صف/سطر واحد، سيتم إرجاع رقم التعريف لآخر صف/سطر تمت إضافته

Example

Insert one row, and return the **ID**.

هنا في المثال

قمنا بإضافة صف/سطر في الجدول customers

وإرجاع رقم التعريف

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("Michelle", "Blue Village")
mycursor.execute(sql, val)

mydb.commit()

print("1 record inserted, ID:", mycursor.lastrowid)
```

استدعاء الدالة cursor() لإنشاء الكائن mycursor

هنا قمنا بطباعة رقم التعريف ID لآخر صف/سطر تمت إضافته

إذا لم يظهر خطأ بعد تشغيلك للملف، فهذا يعني أنه تم إضافة سطر جديد في الجدول customers
وسيتم طباعة الجملة التي تعطينا عدد الأسطر المضافة و رقم التعريف لآخر صف تمت إضافته في الجدول

```
C:\Users\My Name>python demo_mysql_insert_id.py
1 record inserted, ID: 15
```

نتيجة التشغيل

➢ Python MySQL Select From — الجلب والاختيار من جدول في قاعدة بيانات في بايثون

➢ Select From a Table — الاختيار من جدول

To select from a table in **MySQL**, use the "**SELECT**" statement.

لاختيار وجلب البيانات المخزنة في جدول في قاعدة البيانات استخدم الاستعلام SELECT

Example

Select all records from the "**customers**" table, and display the result.

في المثال التالي قمنا بجلب كل البيانات الموجودة في الجدول customers وعرضها

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

كتبنا الاستعلام الذي يمكنّنا من جلب كامل البيانات المخزنة في الجدول customers

الدالة fetchall() ترجّع كل البيانات التي قام الاستعلام بإرجاعها

ثم يتم التخزين في الكائن myresult وهو عبارة عن قائمة list

و كل عنصر في هذه القائمة عبارة عن صف tuple

يمثل سطر في الجدول customers

```
C:\Users\My Name>python demo_mysql_select.py
(1, 'John', 'Highway 21')
(2, 'Peter', 'Lowstreet 27')
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
(6, 'Sandy', 'Ocean blvd 2')
(7, 'Betty', 'Green Grass 1')
(8, 'Richard', 'Sky st 331')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(11, 'Ben', 'Park Lane 38')
(12, 'William', 'Central st 954')
(13, 'Chuck', 'Main Road 989')
(14, 'Viola', 'Sideway 1633')
(15, 'Michelle', 'Blue Village')
```

نتيجة التشغيل

ستحصل على نفس النتيجة إذا قمت بتخزين

نفس البيانات التي أدخلناها في الأمثلة والدروس السابقة

الدالة fetchall() تقوم بترجيع كل الصفوف/السطور

التي ارجعها آخر استعلام تم تنفيذه

Note: We use the **fetchall()** method,

which fetches all rows from the last executed statement.

➢ Selecting Columns

تحديد الأعمدة

To select only some of the columns in a table, use the "**SELECT**" statement followed by the column name(s).

لجلب واختيار بعض الأعمدة في الجدول استخدم الاستعلام **SELECT** متبوعا باسم العامود الذي تريد عرضه

Example

في المثال التالي

Select only the **name** and **address** columns.

قمنا بتحديد الأعمدة التالية:

العامود الذي يحتوي الاسم والعامود الذي يحتوي على العنوان

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

```
C:\Users\My Name>python demo_mysql_select_columns.py
('John', 'Highway 21')
('Peter', 'Lowstreet 27')
('Amy', 'Apple st 652')
('Hannah', 'Mountain 21')
('Michael', 'Valley 345')
('Sandy', 'Ocean blvd 2')
('Betty', 'Green Grass 1')
('Richard', 'Sky st 331')
('Susan', 'One way 98')
('Vicky', 'Yellow Garden 2')
('Ben', 'Park Lane 38')
('William', 'Central st 954')
('Chuck', 'Main Road 989')
('Viola', 'Sideway 1633')
('Michelle', 'Blue Village')
```

نتيجة التشغيل

ستحصل على نفس النتيجة إذا قمت بتخزين نفس البيانات التي أدخلناها في الأمثلة والدروس السابقة

➢ Using the fetchone() Method

استخدام الدالة fetchone()

If you are only interested in one row, you can use the **fetchone()** method.

The **fetchone()** method will return the first row of the result.

إذا أردت إرجاع سطر/صف واحد فقط من الجدول، استخدم هذه الدالة **fetchone()** للتعامل مع البيانات التي تم جلبها

ستقوم الدالة بإرجاع أول صف/سطر من النتيجة

Example

في المثال التالي

Fetch only one row.

قمنا بجلب فقط أول صف/سطر ومن ثم عرض البيانات التي جلبها

باستخدام الدالة **fetchone()**

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchone()

print(myresult)
```

قمنا بإرسال استعلام يرجع أكثر من صف/سطر

ستقوم الدالة fetchone()

بترجيع فقط أول سطر/صف الذي قام بإرجاعه لنا الاستعلام

```
C:\Users\My Name>python demo_mysql_select_fetchone.py
(1, 'John', 'Highway 21')
```

نتيجة التشغيل

➢ Python MySQL Where

الجلب من الجدول ضمن شروط في قاعدة بيانات في بايثون

➢ Select With a Filter

الاختيار ضمن شروط محددة

When selecting records from a table, you can filter the selection by using the "**WHERE**" statement.

عند اختيار وجلب صف من الجدول، يمكنك جلب البيانات ضمن شروط محددة باستخدام الاستعلام **WHERE**

Example

في المثال التالي

Select record(s) where the **address** is "**Park Lane 38**".

قمنا بجلب فقط الصفوف/السطور المخزنة في الجدول

التي تحمل عنوان محددا وهو **Park Lane 38**

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address = 'Park Lane 38'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

```
C:\Users\My Name>python demo_mysql_where.py
(11, 'Ben', 'Park Lane 38')
```

نتيجة التشغيل

➢ Wildcard Characters

البحث عن جزء محدد من البيانات

You can also select the records that starts, includes, or ends with a given letter or phrase.

Use the % to represent wildcard characters.

يمكنك تحديد وجلب الصف/السطر الذي يشمل رمز أو جملة معينة في البداية أو النهاية باستخدام الرمز %

Example

Select records where the address contains the word "**way**".

في المثال التالي

قمنا بجلب فقط الصفوف/السطور المخزنة في الجدول

التي تحتوي على الجزء النصي way

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address Like '%way%'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

```
C:\Users\My Name>python demo_mysql_where_wildcard.py
(1, 'John', 'Highway 21')
(9, 'Susan', 'One way 98')
(14, 'Viola', 'Sideway 1633')
```

نتيجة التشغيل

أتممتَ درسك بنجاح!

تابع التقدّم

روابط قد تساعدك

Check the links below

- **Python and MySQL - Populating our Database and Table**
- **Python and MySQL - Selecting and Getting Data**
- **Python and MySQL - Query Conditions with WHERE and Wildcards**

طبّق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم الثاني والسبعون

قواعد البيانات في بايثون ٣

Python MySQL 3

Python MySQL Order By

الترتيب في قاعدة بيانات في بايثون

Sort the Result

ترتيب النتائج

Use the **ORDER BY** statement to sort the result in ascending or descending order.

The **ORDER BY** keyword sorts the result ascending by default. To sort the result in descending order, use the **DESC** keyword.

استخدم الاستعلام **ORDER BY** لترتيب وفرز النتائج تنازليا أو تصاعديا

الاستعلام **ORDER BY** افتراضيا يقوم بالترتيب تنازليا، وللترتيب تصاعديا استخدم الاستعلام **DESC**

Example

Sort the result alphabetically by **name**.

في المثال التالي

قمنا بترتيب الأسماء أبجديا باستخدام الاستعلام **ORDER BY**

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

```
C:\Users\My Name>python demo_mysql_orderby.py
(3, 'Amy', 'Apple st 652')
(11, 'Ben', 'Park Lane 38')
(7, 'Betty', 'Green Grass 1')
(13, 'Chuck', 'Main Road 989')
(4, 'Hannah', 'Mountain 21')
(1, 'John', 'Highway 21')
(5, 'Michael', 'Valley 345')
(15, 'Michelle', 'Blue Village') (2, 'Peter', 'Lowstreet 27')
(8, 'Richard', 'Sky st 331')
(6, 'Sandy', 'Ocean blvd 2')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(14, 'Viola', 'Sideway 1633')
(12, 'William', 'Central st 954')
```

نتيجة التشغيل

➢ ORDER BY DESC

الترتيب التصاعدي

Use the **DESC** keyword to sort the result in a descending order.

استخدم الاستعلام **DESC** لترتيب النتائج ترتيبا تصاعديا

Example

في المثال التالي

Sort the result reverse alphabetically by **name**.

قمنا بترتيب الأسماء أبجديا بالعكس باستخدام الاستعلام **DESC**

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name DESC"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

```
C:\Users\My Name>python demo_mysql_orderby_desc.py
(12, 'William', 'Central st 954') (14, 'Viola', 'Sideway 1633')
(10, 'Vicky', 'Yellow Garden 2')
(9, 'Susan', 'One way 98')
(6, 'Sandy', 'Ocean blvd 2')
(8, 'Richard', 'Sky st 331')
(2, 'Peter', 'Lowstreet 27')
(15, 'Michelle', 'Blue Village') (5, 'Michael', 'Valley 345')
(1, 'John', 'Highway 21')
(4, 'Hannah', 'Mountain 21')
(13, 'Chuck', 'Main Road 989')
(7, 'Betty', 'Green Grass 1')
(11, 'Ben', 'Park Lane 38')
(3, 'Amy', 'Apple st 652')
```

نتيجة التشغيل

➢ Python MySQL Delete From By — طريقة حذف البيانات في قاعدة بيانات في بايثون

➢ Delete Record — حذف سطر/صف

You can delete records from an existing table by using the "**DELETE FROM**" statement.

تستطيع حذف سطر/صف من جدول في قاعدة البيانات وذلك باستخدام الاستعلام **DELETE FROM**

Example

في المثال التالي

Delete any record where the address is "**Mountain 21**".

قمنا بحذف الصف/السطر

الذي يحتوي على العنوان التالي: **Mountain 21**

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")
```

قمنا بطباعة عدد الأسطر التي تم حذفها من للجدول

إذا لم يظهر خطأ بعد تشغيلك للملف، فهذا يعني أنه تم حذف السطر من الجدول **customers**

وسيتم طباعة الجملة التي تعطينا عدد الأسطر المحذوفة

```
C:\Users\My Name>python demo_mysql_delete.py
1 record(s) deleted
```

نتيجة التشغيل

يجب عليك استدعاء الدالة **commit()** لحفظ التغيرات التي تقوم بها من إضافة أو تعديل أو مسح من الجدول في قاعدة البيانات

Important!: Notice the statement: **mydb.commit()** It is required to make the changes, otherwise no changes are made to the table.

الكلمة الشرطية **WHERE** تحدد الصف الذي يجب حذفه، وعدم كتابتها مع العبارة **DELETE** سيتم حذف الصفوف/الأسطر كاملة

Notice the WHERE clause in the DELETE syntax: The **WHERE** clause specifies which record(s) that should be deleted. If you omit the **WHERE** clause, all records will be deleted!

➢ Python MySQL Drop Table

حذف الجدول من قاعدة بيانات في بايثون

➢ Delete a Table

حذف جدول

You can delete an existing table by using the "**DROP TABLE**" statement.

يمكنك حذف جدول موجود في قاعدة البيانات وذلك باستخدام الاستعلام **DROP TABLE**

Example

في المثال التالي

Delete the table "**customers**".

قمنا بحذف الجدول customers

باستخدام الاستعلام **DROP TABLE**

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DROP TABLE customers"

mycursor.execute(sql)

#If this page was executed with no error(s), you have successfully deleted the "customers" table.
```

```
C:\Users\My Name>python demo_mysql_drop_table.py
```

نتيجة التشغيل

إذا لم يظهر خطأ بعد تشغيلك للملف، فهذا يعني أنه تم حذف الجدول customers بنجاح!

➢ Drop Only if Exist

احذف فقط إذا كان الجدول موجودا

If the table you want to delete is already deleted, or for any other reason does not exist, you can use the **IF EXISTS** keyword to avoid getting an error.

إذا كان الجدول الذي تريد حذفه قد تم حذفه بالفعل، أو لأي سبب الجدول الذي تريد حذفه ليس موجودا

استخدام الاستعلام **IF EXISTS** وذلك لتجنب أي خطأ قد يحدث

Example

Delete the table "customers" if it exists.

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DROP TABLE IF EXISTS customers"

mycursor.execute(sql)

#If this page was executed with no error(s), you have successfully deleted the "customers" table.
```

```
C:\Users\My Name>python demo_mysql_drop_table2.py
```

نتيجة التشغيل

إذا لم يظهر خطأ بعد تشغيلك للملف، فهذا يعني أنه تم حذف الجدول customers بنجاح!

أتممت درسك بنجاح!

واصل التعلّم

روابط قد تهمك للاستفادة فقط

Useful links

- **Python and MySQL - Ordering our Queries and Results**
- **Python and MySQL - Deleting Entries and Dropping Tables**

طبّق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم الثالث والسبعون

قواعد البيانات في بايثون ٤

Python MySQL 4

➢ Python MySQL Update Table

تحديث وتعديل الجدول في قاعدة بيانات في بايثون

➢ Update Table

تحديث جدول

You can update existing records in a table by using the "**UPDATE**" statement.

يمكنك تحديث وتعديل بيانات صفوف/سطور الجدول وذلك باستخدام الاستعلام **UPDATE**

Example

Overwrite the address column from "**Valley 345**" to "**Canyoun 123**".

في المثال التالي

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

قمنا بتحديث وتغيير العنوان Valley 345 إلى Canyoun 123 باستخدام الاستعلام **UPDATE**

قمنا بطباعة عدد الأسطر التي تم التعديل عليها

```
C:\Users\My Name>python demo_mysql_update.py
1 record(s) affected
```

نتيجة التشغيل

Important!: Notice the statement: **mydb.commit()** It is required to make the changes, otherwise no changes are made to the table.

Notice the WHERE clause in the UPDATE syntax: The **WHERE** clause specifies which record or records that should be updated. If you omit the **WHERE** clause, all records will be updated!

➢ Python MySQL Limit

الحد من الجدول في قاعدة بيانات في بايثون

➢ Limit the Result

نتائج محددة

You can limit the number of records returned from the query, by using the "**LIMIT**" statement.

يمكنك الحد من عدد صفوف/سطور الجدول التي يتم إرجاعها من الاستعلام وذلك باستخدام الاستعلام **LIMIT**

Example

في المثال التالي

Select the **5** first records in the "**customers**" table.

قمنا بتحديد فقط 5 أسطر/صفوف من الجدول لعرضها باستخدام الاستعلام **LIMIT**

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers LIMIT 5")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

```
C:\Users\My Name>python demo_mysql_limit.py
(1, 'John', 'Highway 21')
(2, 'Peter', 'Lowstreet 27')
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
```

نتيجة التشغيل

➢ Start From Another Position

البدأ من موقع معين في الجدول

If you want to return five records, starting from the third record, you can use the "**OFFSET**" keyword.

إذا أردت استرجاع فقط خمسة صفوف/أسطر بداية من الصف الثالث يمكنك استخدام الاستعلام **OFFSET**

Example

في المثال التالي

Start from position **3**, and return **5** records.

أردنا أن تكون بداية استرجاع البيانات من السطر الثالث

ونقوم بترجيع فقط خمسة صفوف/أسطر

عن طريق استخدام الاستعلام **OFFSET**

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers LIMIT 5 OFFSET 2")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

```
C:\Users\My Name>python demo_mysql_limit_offset.py
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
(6, 'Sandy', 'Ocean blvd 2')
(7, 'Betty', 'Green Grass 1')
```

نتيجة التشغيل

➢ Python MySQL Join

دمج الجداول في قاعدة بيانات في بايثون

➢ Join Two or More Tables

دمج جدولين أو أكثر

You can combine rows from two or more tables, based on a related column between them, by using a **JOIN** statement.

يمكنك دمج صفوف/أسطر من جدولين أو أكثر بناءا على العامود المشترك بينهم وذلك باستخدام الاستعلام JOIN

Consider you have a "**users**" table and a "**products**" table.

افترض أنه لديك جدولين، جدول للمستخدمين وجدول للمنتجات

Users

```
{ id: 1, name: 'John', fav: 154},
{ id: 2, name: 'Peter', fav: 154},
{ id: 3, name: 'Amy', fav: 155},
{ id: 4, name: 'Hannah', fav:},
{ id: 5, name: 'Michael', fav:}
```

Products

```
{ id: 154, name: 'Chocolate Heaven' },
{ id: 155, name: 'Tasty Lemons' },
{ id: 156, name: 'Vanilla Dreams' }
```

These two tables can be combined by using users' **fav** field and products' **id** field.

يمكنك الدمج بين الجدولين باستخدام عامود النكهة في جدول **users** وعامود رقم التعريف في الجدول **products**

Example

Join users and products to see the name of the users favorite product.

في المثال التالي

دمجنا الجدولين (users و products)

لمعرفة أسماء المستخدمين و نكهتهم المفضلة

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT \
  users.name AS user, \
  products.name AS favorite \
  FROM users \
  INNER JOIN products ON users.fav = products.id"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

```
C:\Users\My Name>python demo_mysql_inner_join.py
('John', 'Chocolate Heaven')
('Peter', 'Chocolate Heaven')
('Amy', 'Tasty Lemon')
```

نتيجة التشغيل

يمكنك استخدام الاستعلام INNER JOIN بدلا من استخدام الاستعلام JOIN كلاهما يعطيان نفس النتيجة

Note: You can use **JOIN** instead of **INNER JOIN**. They will both give you the same result.

➢ LEFT JOIN

الاستعلام LEFT JOIN

In the example above, **Hannah**, and **Michael** were excluded from the result, that is because **INNER JOIN** only shows the records where there is a match.

If you want to show all users, even if they do not have a favorite product, use the **LEFT JOIN** statement:

في المثال السابق، تم استبعاد اثنين من المستخدمين من النتيجة وذلك لأن الاستعلام INNER JOIN يعرض فقط الصفوف/الأسطر التي يكون فيها تطابق، إذا أردت عرض جميع المستخدمين حتى لو لم يكن لهم نكهة مفضلة استخدم الاستعلام LEFT JOIN

Example

Select all users and their favorite product.

في المثال التالي

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT \
  users.name AS user, \
  products.name AS favorite \
  FROM users \
  LEFT JOIN products ON users.fav = products.id"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

تم تحديد جميع المستخدمين من الجدول users ونكهاتهم المفضلة

سواءا كان لديهم نكهة مفضلة أم لا

```
C:\Users\My Name>python demo_mysql_left_join.py
('John', 'Chocolate Heaven')
('Peter', 'Chocolate Heaven')
('Amy', 'Tasty Lemon')
('Hannah', None)
('Michael', None)
```

نتيجة التشغيل

➢ RIGHT JOIN

الاستعلام RIGHT JOIN

If you want to return all products, and the users who have them as their favorite, even if no user have them as their favorite, use the **RIGHT JOIN** statement.

إذا أردت استرجاع كل المنتجات والمستخدمين الذين لديهم هذه المنتجات ونكهاتهم المفضلة، حتى المستخدمين الذين ليس لهم نكهات مفضلة، استخدم الاستعلام RIGHT JOIN

Example

Select all products, and the user(s) who have them as their favourite.

في المثال التالي

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  passwd="mypassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT \
  users.name AS user, \
  products.name AS favorite \
  FROM users \
  RIGHT JOIN products ON users.fav = products.id"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

تم تحديد جميع المنتجات (النكهة المفضلة) من الجدول products والمستخدمين الذين لديهم هذه المنتجات كـ نكهة مفضلة

```
C:\Users\My Name>python demo_mysql_right_join.py
('John', 'Chocolate Heaven')
('Peter', 'Chocolate Heaven')
('Amy', 'Tasty Lemon')
(None, 'Vanilla Dreams')
```

نتيجة التشغيل

رائع!

أتممت درسك الأخير لهذا الأسبوع

وللمزيد من الطرق والتفاصيل للتعامل مع قواعد البيانات قم بزيارة وتصفح موقعهم الرسمي

MySQL Documentation

روابط قد تهمك

Useful links

Python and MySQL - Updating Entries and Limiting Queries

SQL Joins Explained || Joins in SQL || SQL Tutorial

Using MySQL Databases With Python Course

طبّق ما تعلمته في هذا الدرس

ولا تنسى مشاركتنا أكوادك

اليوم الرابع والسبعون

&

اليوم الخامس والسبعون

تحدي الأسبوع (يتم حله ورفعه على Github)

التحدي الأول

١/ اكتب برنامجا يقوم بقراءة كامل الملف النصي، محتوى الملف النصي هو كالتالي: text file

```
What is Python language?

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language.

Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in language such as C++ or Java.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.
```

٢/ ثم قم بكتابة وإلحاق النص التالي على الملف النصي السابق وقم بطباعته وعرضه

```
The best way we learn anything is by practice and exercise questions
```

التحدي الثاني

قم بإنشاء جدول اسمه Employee داخل قاعدة البيانات MyEmployee ويحتوي الجدول على المفتاح الأساسي Primary Key لكل عامود

قم بتعبئة الجدول بالبيانات كما في الجدول التالي:

الاسم الأول FirstName	الاسم الأخير LastName	العمر Age	الجنس Gender	الراتب Salary
Ahmed	Ali	30	Male	10000
Khalid	Muhammad	34	Male	7000
Norah	Saleh	29	Female	7000

١/ بعد ما قمت بإنشاء الجدول وإدخال البيانات قم بعرض كامل محتويات الجدول وطباعته على المحرر الخاص بك أو في موجه الأوامر وصور النتيجة

٢/ قم بعرض فقط الأعمدة التي تحتوي على الاسم الأول والجنس والراتب

٣/ اعرض البيانات كاملة مرتبة ترتيبا أبجديا عكسيا بواسطة الاسم الأول

٤/ احذف الصف الذي يحتوي على العمر 34 ثم قم بعرض كامل البيانات بعد الحذف

تحدّي إضافي - لستَ مُلزمًا بحله -

اكتب برنامجا يقوم بقراءة الملف سطر بسطر ثم يتم تخزين المحتوى في قائمة، يمكنك استخدام الدالة readlines() ابحث عنها

موفق دومًا

انتهت سلسلة الدروس

في الأسابيع القادمة تنتظرك تحديات أكبر وهو المشروع النهائي .. كُن مستعدا