

Data Processing and Analysis in Python

Lecture 12

Databases



UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

DR. ADAM LEE

Different Ways to Execute Python Programs

- Use python console (e.g. IDLE), but with limited functionality:

```
>>> print("Hello!!!")  
Hello!!!
```

- Call .py program via python interpreter from command line:

```
% python hello.py
```

- Make .py directly executable, with additional header lines:

```
#!/usr/bin/env python
```

- Use interactive console (e.g. IPython, Jupyter):

```
In [1]: print("Hello!!!")  
Out [1]: Hello!!!  
In [2]: %run hello.py
```



UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

Relational Databases

- A collection of data items organized as a set of formally described relations from which data can be accessed easily
- Use **SQL (Structured Query Language)** to define and manipulate relational databases
- Each database consists of a number of **relations** (a.k.a. tables) and each relation has its own **primary key (PK)**
- Relations are connected by **foreign keys (FKs)**



UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

SQL

- CREATE TABLE ...
- INSERT INTO <table> VALUES ...
- SELECT ... FROM <table> ...
- UPDATE <table> SET ...
- DELETE FROM <table> WHERE ...
- DROP TABLE ...



UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

Relational Database Management Systems

RDBMS	Vendor	First release	Latest release
<u>Db2</u>	<u>IBM</u>	1993	2019
<u>Access</u>	<u>Microsoft</u>	1992	2020
<u>SQL Server</u>	<u>Microsoft</u>	1989	2019
<u>MySQL</u>	<u>Oracle</u>	1995	2020
<u>Oracle</u>	<u>Oracle</u>	1979	2019
<u>PostgreSQL</u>	<u>PostgreSQL</u>	1996	2020
<u>SQL Anywhere</u>	<u>SAP</u>	1992	2015
<u>Adaptive Server Enterprise (ASE)</u>	<u>SAP</u>	1987	2020

MARYLAND

Optional: Install MySQL on Microsoft Windows

- Download and execute MySQL Installer from <https://dev.mysql.com/downloads/installer/>
- Determine the setup type:
 - Developer Default: Server and other tools, such as Workbench
 - Server Only: Server without other products
 - Custom: Enables you to select Server and other products
- To install the server instance:
 - Standalone Server / Classic Replication (default)
 - InnoDB cluster: Configure on the local host or add a new server instance to an existing InnoDB Cluster
- Complete the configuration process



UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

Optional: Install MySQL on MacOS

- MacOS versions that the MySQL server supports
<https://mysql.com/support/supportedplatforms/database.html>
- General Notes on Installing MySQL on MacOS
<https://dev.mysql.com/doc/refman/8.0/en/osx-installation-notes.html>
- MySQL for MacOS is available in different forms:
 - Native Package Installer (i.e. DMG) to walk you through the installation
<https://dev.mysql.com/doc/refman/8.0/en/osx-installation-pkg.html>
 - Compressed TAR archive, you do not need administrator privileges
<https://dev.mysql.com/doc/refman/8.0/en/binary-installation.html>
 - Package Installer, to simplify the management of installation
<https://dev.mysql.com/doc/refman/8.0/en/osx-installation-launchd.html>



UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

Install Anaconda

- Anaconda Individual Edition

<https://anaconda.com/products/individual>

Anaconda Installers

Windows 

MacOS 

Python 3.8

64-Bit Graphical Installer (466 MB)

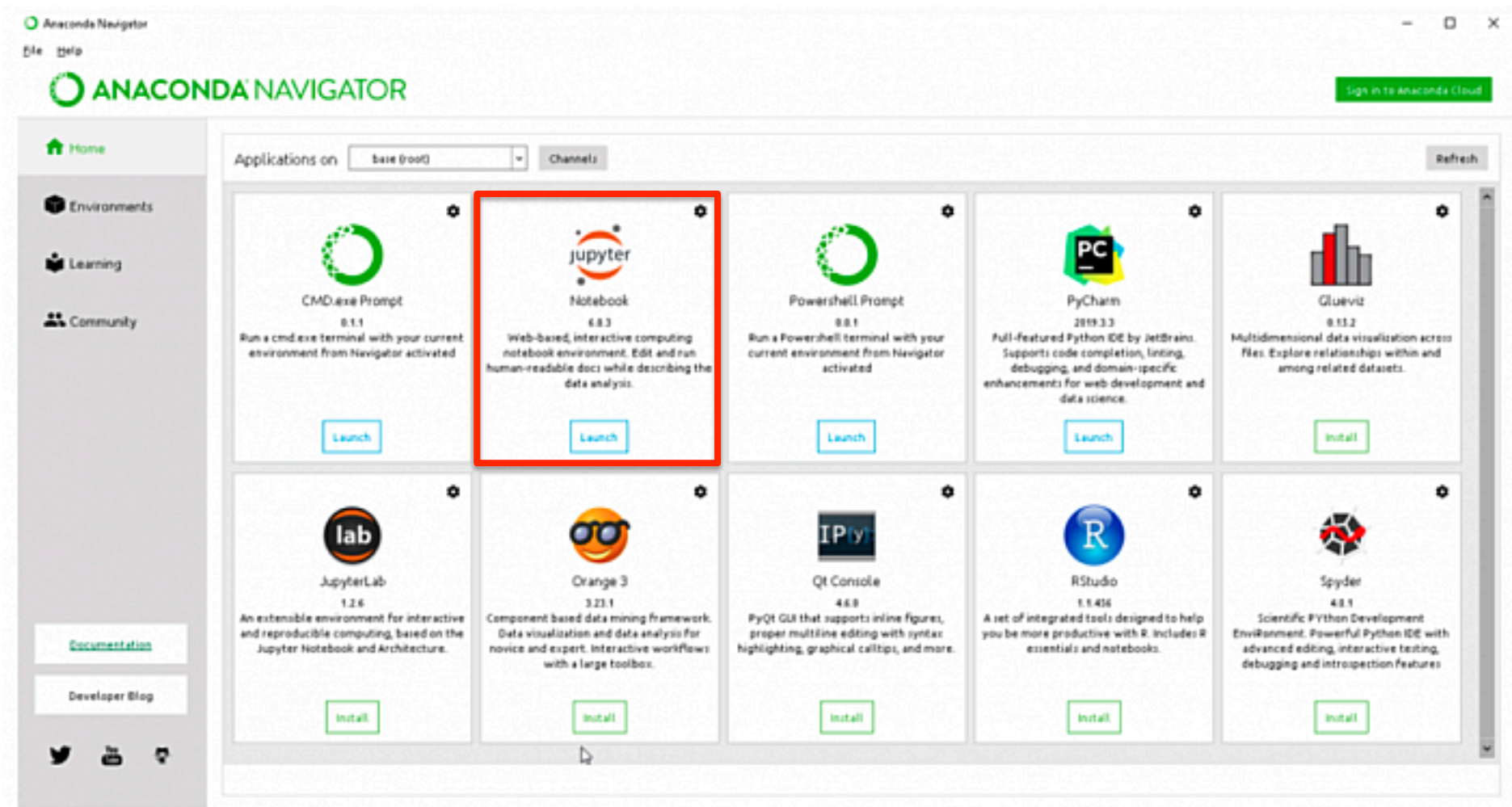
32-Bit Graphical Installer (397 MB)

Python 3.8

64-Bit Graphical Installer (462 MB)

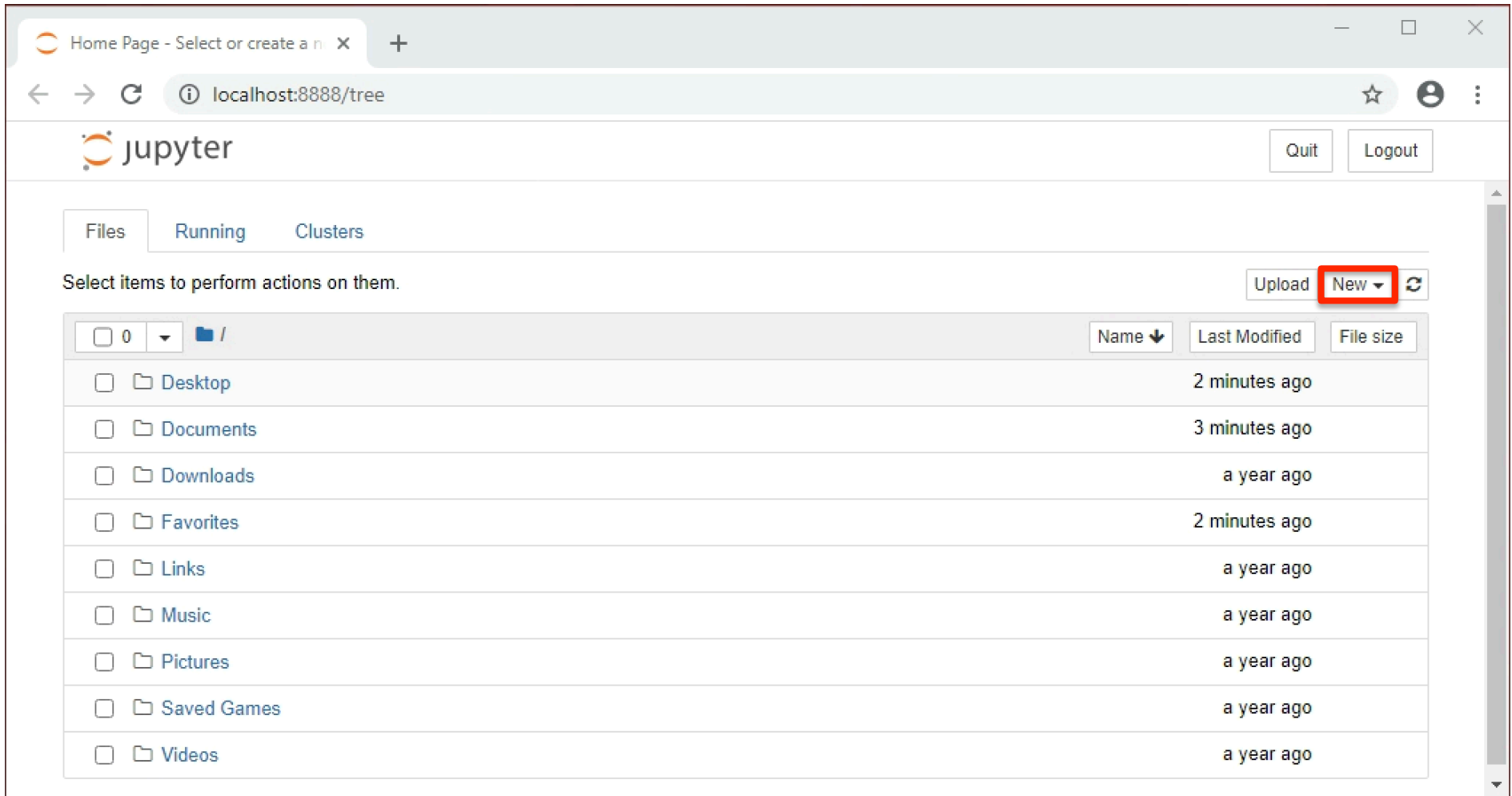
64-Bit Command Line Installer (454 MB)

Anaconda Navigator and Jupyter Notebook



UNIVERSITY OF
MARYLAND

Jupyter Notebook on vSmith



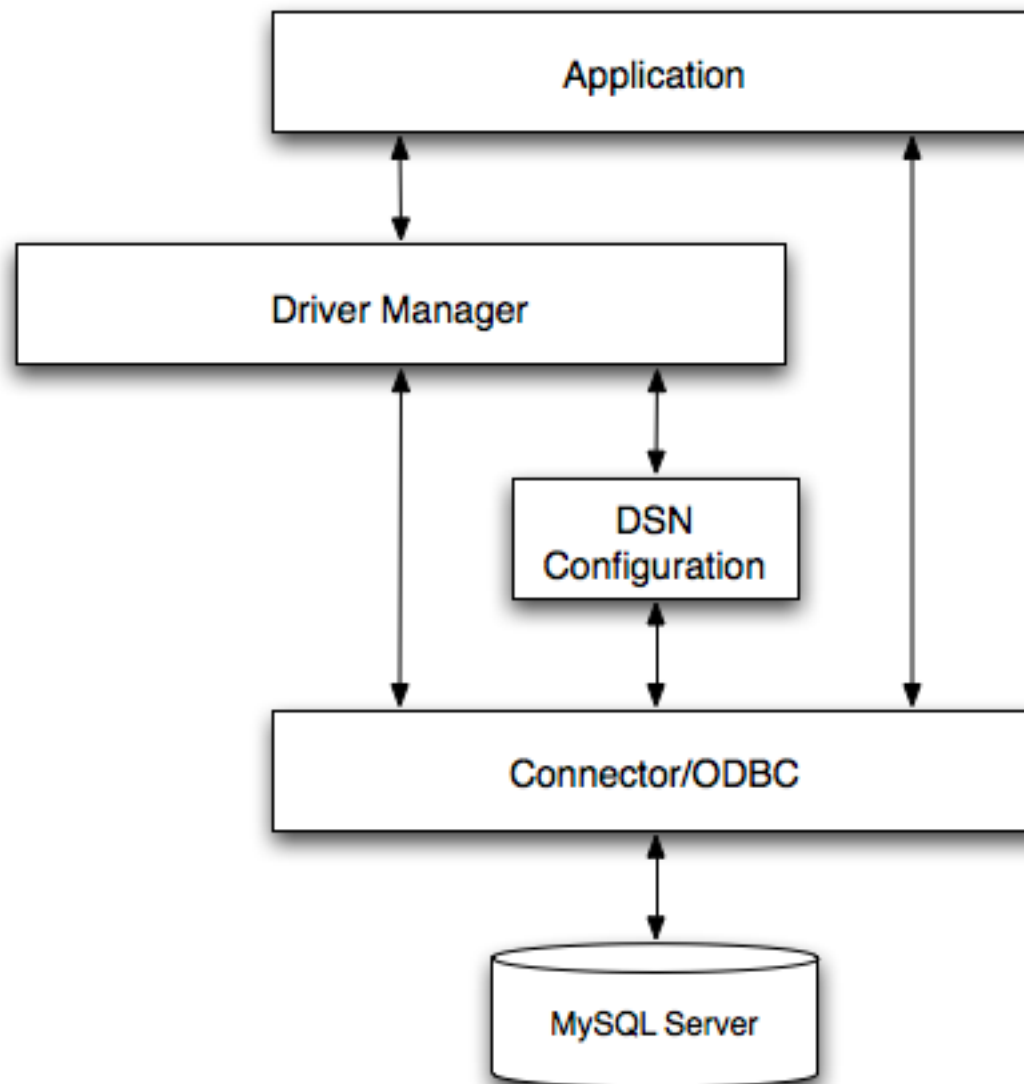
The screenshot shows the Jupyter Notebook web interface in a browser window. The address bar indicates the URL is `localhost:8888/tree`. The interface includes a header with the Jupyter logo, a "Quit" button, and a "Logout" button. Below the header, there are tabs for "Files", "Running", and "Clusters". The "Files" tab is active, displaying a file browser view. At the top of the file browser, there is a prompt "Select items to perform actions on them." and buttons for "Upload", "New" (highlighted with a red box), and a refresh icon. Below this, a table lists the contents of the current directory. The table has columns for "Name", "Last Modified", and "File size". The contents include a list of standard desktop folders: Desktop, Documents, Downloads, Favorites, Links, Music, Pictures, Saved Games, and Videos. Each item has a checkbox on the left and a timestamp on the right.

	Name	Last Modified	File size
<input type="checkbox"/>	/		
<input type="checkbox"/>	Desktop	2 minutes ago	
<input type="checkbox"/>	Documents	3 minutes ago	
<input type="checkbox"/>	Downloads	a year ago	
<input type="checkbox"/>	Favorites	2 minutes ago	
<input type="checkbox"/>	Links	a year ago	
<input type="checkbox"/>	Music	a year ago	
<input type="checkbox"/>	Pictures	a year ago	
<input type="checkbox"/>	Saved Games	a year ago	
<input type="checkbox"/>	Videos	a year ago	



UNIVERSITY OF
MARYLAND

MySQL Connector Architecture



Step 1: Establish a Connection

- Allocate Environment Handle
- Set ODBC Version
- Allocate Connection Handle
- Connect to MySQL Server
- Set Optional Connection Attributes

Step 2: Initialize the Statement

- Allocate Statement Handle
- Set Optional Statement Attributes

Step 3: Execute SQL Statement

- Prepare the SQL statement
- Execute the SQL statement or execute it directly without prepare

Statement Type?

SELECT / SHOW / Catalog API

DELETE / UPDATE / INSERT

Other

Step 4: Fetch Results

- Get Number of Columns
- Get Column Information
- Fetch Rows
- Get the data to buffers

Step 4: Fetch Results

- Get Number of rows affected

Step 5: Transaction

- Perform commit or rollback

Step 6: Disconnect

- Disconnect the connection
- Free Connection Handle
- Free Environment Handle



Interact with MySQL Server

Step 0: Configure the connection

import ...

Step 1: Establish a connection to MySQL server

connect()

Step 2: Initialize operations in SQL statement

cursor()

Step 3: Execute the SQL statement

execute(...)

Step 4: Fetch results

fetch...()

Step 5: Perform transactions

close()

Step 6: Disconnect from the server

close()



UNIVERSITY OF
MARYLAND

MySQL Connector Example

```
!pip install mysql-connector-python
import mysql.connector
conn = mysql.connector.connect(
    host="bmgt406.rhsmith.umd.edu", user="budt703",
    password="budt703", database="budt703_db")
curs = conn.cursor()
curs.execute("SELECT * FROM Employee_T")
for row in curs.fetchall() :
    print(row)
curs.close()
conn.close()

('000-00-000', 'Laura Ellenburg', '5342 Picklied Trout Lane', 'Nashville', 'TN', '38010', None, None, '454-56-768')
('123-44-345', 'Phil Morris', '2134 Hilltop Rd', 'Knoxville', 'TN', '37920', None, None, '454-56-768')
('334-45-667', 'Lawrence Haley', '5970 Spring Crest Rd', 'Nashville', 'TN', '54545', None, None, '454-56-768')
('454-56-768', 'Robert Lewis', '17834 Deerfield Ln', 'Knoxville', 'TN', '55555', None, None, '123-44-345')
('559-55-585', 'Mary Smith', '75 Jane Lane', 'Clearwater', 'FL', '33879', None, None, '334-45-667')
```

Choose Database Server Connector

- `import mysql.connector`
`conn = mysql.connector.connect(...)`
- `import mysqldb`
`conn = mysqldb.connect(...)`
- `import sqlite3`
`conn = sqlite3.connect(...)`
- `import pyodbc`
`conn = pyodbc.connect(...)`
- `import cx_Oracle`
`conn = cx_Oracle.connect(...)`
- ...



UNIVERSITY OF
MARYLAND

Microsoft SQL Server Connector Example

```
import pyodbc
server = "doitsqlx.rhsmith.umd.edu,9703"
database = "BUDT703_DB_Student_nnn"
username = "BUDT703_Student_nnn"
password = "BUDT703_Student_nnn"
conn = pyodbc.connect("DRIVER={ODBC Driver 17  
for SQL Server};SERVER=" + server +  
";DATABASE=" + database + ";UID=" + username +  
";PWD=" + password)
curs = conn.cursor()
curs.execute("SELECT * FROM Employee_T")
for row in curs.fetchall() :
    print(row)
curs.close()
conn.close()
```



UNIVERSITY OF
MARYLAND

SQLite Connector Example

- Database in memory:

```
import sqlite3  
conn = sqlite3.connect(":memory:")  
...  
conn.close()
```

- Database in file:

```
import sqlite3  
conn = sqlite3.connect("personal.db")  
...  
conn.close()
```



UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

Error Exception

try:

```
conn = mysql.connector.connect(...)
curs = conn.cursor()
curs.execute(...)
...
curs.close()
conn.close()
```

except mysql.connector.Error as err:

```
    print("Error:", err)
```

else:

```
    ...
```

finally:

```
    print("Bye!")
```



UNIVERSITY OF
MARYLAND

Python MySQL CREATE TABLE

- Check if a table exist:

```
curs.execute("SHOW TABLES")  
for row in curs:  
    print(row)
```

- Create table:

```
curs.execute("""CREATE TABLE Student(  
studentId CHAR(3) NOT NULL,  
studentName VARCHAR(50),  
CONSTRAINT pk_Student_studentId PRIMARY KEY  
(studentId) )""")
```



UNIVERSITY OF
MARYLAND

Python MySQL INSERT INTO

```
curs.execute("INSERT INTO Student VALUES  
('S01','First Last')")  
conn.commit()
```

```
curs.execute("INSERT INTO Student VALUES (%s,  
%s)" % ("S02","Second Last"))  
conn.commit()
```

```
try:  
    sql = "INSERT INTO Student VALUES (%s,%s)"  
    val = ("S03","Third Last")  
    curs.execute(sql % val)  
    conn.commit()  
except:  
    conn.rollback()
```



UNIVERSITY OF
MARYLAND

Python MySQL SELECT FROM

- `fetchall()`: return all rows from the last executed statement

```
curs.execute("SELECT * FROM Student")
for row in curs.fetchall():
    print(row)
```

- `fetchone()`: return the first row from remaining of the last executed statement

```
curs.execute("SELECT * FROM Student")
print(curs.fetchone())
...
```

- `rowcount`: number of rows fetched

```
print(curs.rowcount)
```



UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

Python MySQL UPDATE SET

```
try:
    curs.execute("UPDATE Student SET
studentName = 'Bob Smith' WHERE studentId =
'S03'")
    conn.commit()
except:
    conn.rollback()
```



UNIVERSITY OF
MARYLAND

Python MySQL DELETE FROM

```
try:
    curs.execute("DELETE FROM Student WHERE
studentId = 'S03'")
    conn.commit()
except:
    conn.rollback()
```



UNIVERSITY OF
MARYLAND

Python MySQL DROP TABLE

- Delete a table:

```
curs.execute("DROP TABLE Student")
```

- Delete table only if exists:

```
curs.execute("DROP TABLE IF EXISTS Student")
```



UNIVERSITY OF
MARYLAND