

Working with MySQL

MySQL basics

starting and stopping the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and table

Using MySQLdb

inserting records with Python script
scanning files and grabbing the headers
filling the table in the database

1 MySQL basics

starting and stopping the daemon
running the monitor mysql

2 Database for Python scripts

problem statement
create database and table

3 Using MySQLdb

inserting records with Python script
scanning files and grabbing the headers
filling the table in the database

MCS 275 Lecture 25
Programming Tools and File Management
Jan Verschelde, 10 March 2010

Working with MySQL

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

1 MySQL basics

starting and stopping the daemon

running the monitor mysql

2 Database for Python scripts

problem statement

create database and table

3 Using MySQLdb

inserting records with Python script

scanning files and grabbing the headers

filling the table in the database

10 Mar 2010

Starting and Stopping the Daemon

as root on Mac OS X

It may be that MySQL is started at boot time.

Otherwise:

```
$ sudo mysqld_safe
```

```
Starting mysqld daemon with databases  
from /usr/local/mysql/data
```

Shutting the MySQL server down:

```
$ sudo mysqladmin shutdown
```

```
STOPPING server from pid file  
/usr/local/mysql/data/ambiorix.local.pid  
080304 21:33:10  mysqld ended
```

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

10 Mar 2010

Starting and Stopping the Daemon

as root on Mac OS X

MySQL basics

starting and stopping
the daemonrunning the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

It may be that MySQL is started at boot time.

Otherwise:

```
$ sudo mysqld_safe
```

Starting mysqld daemon with databases
from /usr/local/mysql/data

Shutting the MySQL server down:

```
$ sudo mysqladmin shutdown
```

```
STOPPING server from pid file  
/usr/local/mysql/data/ambiorix.local.pid  
080304 21:33:10  mysqld ended
```

Starting and Stopping the Daemon

as root on Mac OS X

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

It may be that MySQL is started at boot time.

Otherwise:

```
$ sudo mysqld_safe
```

Starting mysqld daemon with databases
from /usr/local/mysql/data

Shutting the MySQL server down:

```
$ sudo mysqladmin shutdown
```

```
STOPPING server from pid file  
/usr/local/mysql/data/ambiorix.local.pid  
080304 21:33:10  mysqld ended
```

Creating and Deleting Databases

with `mysqladmin`

MySQL basics

starting and stopping
the daemon

running the monitor
`mysql`

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

To create a database we use the `create` command with `mysqladmin`:

```
$ sudo mysqladmin create mydb
```

To delete the database we use the `drop` command with `mysqladmin`:

```
$ sudo mysqladmin drop mydb
```

Dropping the database is potentially a very bad thing.
Any data stored in the database will be destroyed.

```
Do you really want to drop the 'mydb' database [y/N]
Database "mydb" dropped
```

Creating and Deleting Databases

with `mysqladmin`

MySQL basics

starting and stopping
the daemon

running the monitor
`mysql`

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

To create a database we use the `create` command with `mysqladmin`:

```
$ sudo mysqladmin create mydb
```

To delete the database we use the `drop` command with `mysqladmin`:

```
$ sudo mysqladmin drop mydb
```

Dropping the database is potentially a very bad thing.
Any data stored in the database will be destroyed.

```
Do you really want to drop the 'mydb' database [y/N]
Database "mydb" dropped
```

Creating and Deleting Databases

with `mysqladmin`

MySQL basics

starting and stopping
the daemon

running the monitor
`mysql`

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

To create a database we use the `create` command with `mysqladmin`:

```
$ sudo mysqladmin create mydb
```

To delete the database we use the `drop` command with `mysqladmin`:

```
$ sudo mysqladmin drop mydb
```

Dropping the database is potentially a very bad thing.
Any data stored in the database will be destroyed.

```
Do you really want to drop the 'mydb' database [y/N]
Database "mydb" dropped
```

Creating and Deleting Databases

with `mysqladmin`

MySQL basics

starting and stopping
the daemon

running the monitor
`mysql`

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

To create a database we use the `create` command with `mysqladmin`:

```
$ sudo mysqladmin create mydb
```

To delete the database we use the `drop` command with `mysqladmin`:

```
$ sudo mysqladmin drop mydb
```

Dropping the database is potentially a very bad thing.
Any data stored in the database will be destroyed.

```
Do you really want to drop the 'mydb' database [y/N]
Database "mydb" dropped
```

Working with MySQL

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

1 MySQL basics

starting and stopping the daemon
running the monitor mysql

2 Database for Python scripts

problem statement
create database and table

3 Using MySQLdb

inserting records with Python script
scanning files and grabbing the headers
filling the table in the database

10 Mar 2010

MySQL basics

starting and stopping
the daemonrunning the monitor
mysqlDatabase for
Python scriptsproblem statement
create database and
tableUsing
MySQLdbinserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

Running the Monitor `mysql`

command line interface

```
$ sudo mysql
```

```
Welcome to the MySQL monitor.  Commands end with ;  
Your MySQL connection id is 4  
Server version: 5.0.51a MySQL Community Server (GPL)
```

```
Type 'help;' or '\h' for help. Type '\c' to clear t
```

```
mysql>
```

```
Leaving:
```

```
mysql> exit
```

```
Bye
```

Command line `mysql` is very useful to try commands *quickly* while writing a script.

10 Mar 2010

MySQL basics

starting and stopping
the daemonrunning the monitor
mysqlDatabase for
Python scriptsproblem statement
create database and
tableUsing
MySQLdbinserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

Running the Monitor `mysql`

command line interface

```
$ sudo mysql
```

```
Welcome to the MySQL monitor.  Commands end with ;  
Your MySQL connection id is 4  
Server version: 5.0.51a MySQL Community Server (GPL)
```

```
Type 'help;' or '\h' for help. Type '\c' to clear t
```

```
mysql>
```

Leaving:

```
mysql> exit
```

```
Bye
```

Command line `mysql` is very useful to try commands *quickly* while writing a script.

Basic `mysql` Commands

abbreviated syntax and description

command syntax	description
use <dbname>	make database current
create table <name> <field(s)>	create a table
drop table <name>	delete a table

To change a table, we use *queries*:

command syntax	description
select <field(s)> from <table>	retrieve records
insert into <table> <values>	insert records
delete from <table>	delete records
update <table> set <values>	update records

In addition: **where** <criteria> **order by** <field> **ASC** | **DSC**

Basic `mysql` Commands

abbreviated syntax and description

command syntax	description
use <dbname>	make database current
create table <name> <field(s)>	create a table
drop table <name>	delete a table

To change a table, we use *queries*:

command syntax	description
select <field(s)> from <table>	retrieve records
insert into <table> <values>	insert records
delete from <table>	delete records
update <table> set <values>	update records

In addition: **where** <criteria> **order by** <field> **ASC** | **DSC**

Basic `mysql` Commands

abbreviated syntax and description

command syntax	description
use <dbname>	make database current
create table <name> <field(s)>	create a table
drop table <name>	delete a table

To change a table, we use *queries*:

command syntax	description
select <field(s)> from <table>	retrieve records
insert into <table> <values>	insert records
delete from <table>	delete records
update <table> set <values>	update records

In addition: **where** <criteria> **order by** <field> **ASC** | **DSC**

Basic `mysql` Commands

abbreviated syntax and description

command syntax	description
use <dbname>	make database current
create table <name> <field(s)>	create a table
drop table <name>	delete a table

To change a table, we use *queries*:

command syntax	description
select <field(s)> from <table>	retrieve records
insert into <table> <values>	insert records
delete from <table>	delete records
update <table> set <values>	update records

In addition: **where** <criteria> **order by** <field> **ASC** | **DSC**

Basic `mysql` Commands

abbreviated syntax and description

command syntax	description
use <dbname>	make database current
create table <name> <field(s)>	create a table
drop table <name>	delete a table

To change a table, we use *queries*:

command syntax	description
select <field(s)> from <table>	retrieve records
insert into <table> <values>	insert records
delete from <table>	delete records
update <table> set <values>	update records

In addition: **where** <criteria> **order by** <field> **ASC** | **DSC**

Basic `mysql` Commands

abbreviated syntax and description

command syntax	description
use <dbname>	make database current
create table <name> <field(s)>	create a table
drop table <name>	delete a table

To change a table, we use *queries*:

command syntax	description
select <field(s)> from <table>	retrieve records
insert into <table> <values>	insert records
delete from <table>	delete records
update <table> set <values>	update records

In addition: **where** <criteria> **order by** <field> **ASC** | **DSC**

Basic `mysql` Commands

abbreviated syntax and description

command syntax	description
use <dbname>	make database current
create table <name> <field(s)>	create a table
drop table <name>	delete a table

To change a table, we use *queries*:

command syntax	description
select <field(s)> from <table>	retrieve records
insert into <table> <values>	insert records
delete from <table>	delete records
update <table> set <values>	update records

In addition: **where** <criteria> **order by** <field> **ASC** | **DSC**

Working with MySQL

MySQL basics

starting and stopping the daemon

running the monitor mysql

Database for Python scripts

problem statement

create database and table

Using MySQLdb

inserting records with Python script

scanning files and grabbing the headers
filling the table in the database

1 MySQL basics

starting and stopping the daemon
running the monitor mysql

2 Database for Python scripts

problem statement
create database and table

3 Using MySQLdb

inserting records with Python script
scanning files and grabbing the headers
filling the table in the database

Database to manage our Scripts

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

So far, there are over 70 Python scripts
posted at the course web site.

The scripts are listed chronologically
grouped along the lectures . . .
but then, there are also the scripts
for the projects and quizzes.

Goal: build a systematical catalog.
→ sort the scripts in several ways

Database to manage our Scripts

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement

create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

So far, there are over 70 Python scripts
posted at the course web site.

The scripts are listed chronologically
grouped along the lectures . . .
but then, there are also the scripts
for the projects and quizzes.

Goal: build a systematical catalog.
→ sort the scripts in several ways

Database to manage our Scripts

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement

create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

So far, there are over 70 Python scripts
posted at the course web site.

The scripts are listed chronologically
grouped along the lectures . . .
but then, there are also the scripts
for the projects and quizzes.

Goal: build a systematical catalog.
→ sort the scripts in several ways

Database to manage our Scripts

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

So far, there are over 70 Python scripts posted at the course web site.

The scripts are listed chronologically grouped along the lectures . . . but then, there are also the scripts for the projects and quizzes.

Goal: build a systematical catalog.
→ sort the scripts in several ways

Working with MySQL

MySQL basics

starting and stopping the daemon

running the monitor mysql

Database for Python scripts

problem statement

create database and table

Using MySQLdb

inserting records with Python script

scanning files and grabbing the headers

filling the table in the database

1 MySQL basics

starting and stopping the daemon
running the monitor mysql

2 Database for Python scripts

problem statement
create database and table

3 Using MySQLdb

inserting records with Python script
scanning files and grabbing the headers
filling the table in the database

10 Mar 2010

MySQL basics

starting and stopping
the daemonrunning the monitor
mysqlDatabase for
Python scripts

problem statement

**create database and
table**Using
MySQLdbinserting records with
Python scriptscanning files and
grabbing the headersfilling the table in the
database

Creating Database and Table

using mysqladmin and mysql

```
$ sudo mysqladmin create OurPyFiles
```

The database `OurPyFiles` has one table: `scripts`.

For every script, we have a type (L, P, or Q),
a number, a date, and a file name.

```
mysql> use OurPyFiles
Database changed
mysql> create table scripts
  -> (t CHAR(1), n INT, d DATE, f CHAR(20));
Query OK, 0 rows affected (0.00 sec)
```

mysql commands are closed with a semicolon ;

Creating Database and Table

using mysqladmin and mysql

```
$ sudo mysqladmin create OurPyFiles
```

The database `OurPyFiles` has one table: `scripts`.

For every script, we have a type (L, P, or Q),
a number, a date, and a file name.

```
mysql> use OurPyFiles
Database changed
mysql> create table scripts
  -> (t CHAR(1), n INT, d DATE, f CHAR(20));
Query OK, 0 rows affected (0.00 sec)
```

mysql commands are closed with a semicolon ;

10 Mar 2010

MySQL basics

starting and stopping
the daemonrunning the monitor
mysqlDatabase for
Python scriptsproblem statement
create database and
tableUsing
MySQLdbinserting records with
Python scriptscanning files and
grabbing the headersfilling the table in the
database

Creating Database and Table

using mysqladmin and mysql

```
$ sudo mysqladmin create OurPyFiles
```

The database `OurPyFiles` has one table: `scripts`.
For every script, we have a type (L, P, or Q),
a number, a date, and a file name.

```
mysql> use OurPyFiles
Database changed
mysql> create table scripts
  -> (t CHAR(1), n INT, d DATE, f CHAR(20));
Query OK, 0 rows affected (0.00 sec)
```

mysql commands are closed with a semicolon ;

Creating Database and Table

using mysqladmin and mysql

```
$ sudo mysqladmin create OurPyFiles
```

The database `OurPyFiles` has one table: `scripts`.

For every script, we have a type (L, P, or Q),
a number, a date, and a file name.

```
mysql> use OurPyFiles
Database changed
mysql> create table scripts
  -> (t CHAR(1), n INT, d DATE, f CHAR(20));
Query OK, 0 rows affected (0.00 sec)
```

mysql commands are closed with a semicolon ;

Data Types in MySQL

Most commonly used data types:

numeric types	description
INT	integer in $[-2^{31}, 2^{31} - 1]$
SMALLINT	2-byte integer
FLOAT	floating-point number

date and time	description
YEAR	year as <code>yyyy</code>
DATE	date in format <code>yyyy-mm-dd</code>
TIME	time in format <code>hh:mm:ss</code>
DATETIME	<code>yyyy-mm-dd hh:mm:ss</code>
TIMESTAMP	date expressed in seconds

string types	description
CHAR(size)	string of length <code>size</code>
TEXT	strings of unlimited length

MySQL basics

starting and stopping
the daemon

running the monitor
`mysql`

Database for Python scripts

problem statement

create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

Data Types in MySQL

Most commonly used data types:

numeric types	description
INT	integer in $[-2^{31}, 2^{31} - 1]$
SMALLINT	2-byte integer
FLOAT	floating-point number

date and time	description
YEAR	year as <i>yyyy</i>
DATE	date in format <i>yyyy-mm-dd</i>
TIME	time in format <i>hh:mm:ss</i>
DATETIME	<i>yyyy-mm-dd hh:mm:ss</i>
TIMESTAMP	date expressed in seconds

string types	description
CHAR(size)	string of length <i>size</i>
TEXT	strings of unlimited length

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement

create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

Data Types in MySQL

Most commonly used data types:

numeric types	description
INT	integer in $[-2^{31}, 2^{31} - 1]$
SMALLINT	2-byte integer
FLOAT	floating-point number

date and time	description
YEAR	year as <i>yyyy</i>
DATE	date in format <i>yyyy-mm-dd</i>
TIME	time in format <i>hh:mm:ss</i>
DATETIME	<i>yyyy-mm-dd hh:mm:ss</i>
TIMESTAMP	date expressed in seconds

string types	description
CHAR(size)	string of length <i>size</i>
TEXT	strings of unlimited length

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement

create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

Seeing Tables and their Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
**create database and
table**

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
mysql> show tables;
```

```
+-----+
| Tables_in_ourpyfiles |
+-----+
| scripts                |
+-----+
1 row in set (0.00 sec)
```

```
mysql> explain scripts;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| t     | char(1)   | YES  |     | NULL    |       |
| n     | int(11)   | YES  |     | NULL    |       |
| d     | date      | YES  |     | NULL    |       |
| f     | char(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

10 Mar 2010

Seeing Tables and their Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
**create database and
table**

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
mysql> show tables;
```

```
+-----+
| Tables_in_ourpyfiles |
+-----+
| scripts                |
+-----+
1 row in set (0.00 sec)
```

```
mysql> explain scripts;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| t     | char(1)       | YES  |     | NULL    |       |
| n     | int(11)       | YES  |     | NULL    |       |
| d     | date          | YES  |     | NULL    |       |
| f     | char(20)      | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Working with MySQL

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers
filling the table in the
database

1 MySQL basics

starting and stopping the daemon
running the monitor mysql

2 Database for Python scripts

problem statement
create database and table

3 Using MySQLdb

inserting records with Python script
scanning files and grabbing the headers
filling the table in the database

10 Mar 2010

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for
Python scripts

problem statement
create database and
table

Using
MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers
filling the table in the
database

Inserting Records

with a Python script

The table will contain over 70 records.

Entering the data manually is tedious
and may lead to errors.

Therefore, we will use the module `MySQLdb`:

```
$ sudo python
Python 2.5.1 (r251:54869, Apr 18 2007, 22:08:04
[GCC 4.0.1 (Apple Computer, Inc. build 5367)] c
Type "help", "copyright", "credits" or "license
>>> import MySQLdb
```

Two tasks:

- 1 grab the headers of each .py file
- 2 insert header fields into table

10 Mar 2010

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for
Python scripts

problem statement
create database and
table

Using
MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers
filling the table in the
database

Inserting Records

with a Python script

The table will contain over 70 records.

Entering the data manually is tedious
and may lead to errors.

Therefore, we will use the module `MySQLdb`:

```
$ sudo python
Python 2.5.1 (r251:54869, Apr 18 2007, 22:08:04
[GCC 4.0.1 (Apple Computer, Inc. build 5367)] c
Type "help", "copyright", "credits" or "license
>>> import MySQLdb
```

Two tasks:

- 1 grab the headers of each .py file
- 2 insert header fields into table

10 Mar 2010

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for
Python scripts

problem statement
create database and
table

Using
MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers
filling the table in the
database

Inserting Records

with a Python script

The table will contain over 70 records.

Entering the data manually is tedious
and may lead to errors.

Therefore, we will use the module `MySQLdb`:

```
$ sudo python
Python 2.5.1 (r251:54869, Apr 18 2007, 22:08:04)
[GCC 4.0.1 (Apple Computer, Inc. build 5367)] c
Type "help", "copyright", "credits" or "license"
>>> import MySQLdb
```

Two tasks:

- 1 grab the headers of each .py file
- 2 insert header fields into table

10 Mar 2010

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for
Python scripts

problem statement
create database and
table

Using
MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers
filling the table in the
database

Inserting Records

with a Python script

The table will contain over 70 records.

Entering the data manually is tedious
and may lead to errors.

Therefore, we will use the module `MySQLdb`:

```
$ sudo python
Python 2.5.1 (r251:54869, Apr 18 2007, 22:08:04)
[GCC 4.0.1 (Apple Computer, Inc. build 5367)] c
Type "help", "copyright", "credits" or "license"
>>> import MySQLdb
```

Two tasks:

- 1 grab the headers of each .py file
- 2 insert header fields into table

Using the `os` Module

to scan files

All Python scripts are in some directory on disk.

With the `os` module we write utilities
which are independent of the operating system.

Commands we will use:

- `os.listdir(<directory>)` returns a list of strings of names of files and directories in `<directory>`
- `os.getcwd()` returns the path name of the current working directory
- `os.chdir(<path name>)` changes the current working directory to the `<path name>`.

Using the `os` Module

to scan files

All Python scripts are in some directory on disk.

With the `os` module we write utilities
which are independent of the operating system.

Commands we will use:

- `os.listdir(<directory>)` returns a list of strings of names of files and directories in `<directory>`
- `os.getcwd()` returns the path name of the current working directory
- `os.chdir(<path name>)` changes the current working directory to the `<path name>`.

Using the `os` Module

to scan files

All Python scripts are in some directory on disk.

With the `os` module we write utilities
which are independent of the operating system.

Commands we will use:

- `os.listdir(<directory>)` returns a list of strings of names of files and directories in `<directory>`
- `os.getcwd()` returns the path name of the current working directory
- `os.chdir(<path name>)` changes the current working directory to the `<path name>`.

Using the `os` Module

to scan files

All Python scripts are in some directory on disk.

With the `os` module we write utilities
which are independent of the operating system.

Commands we will use:

- `os.listdir(<directory>)` returns a list of strings of names of files and directories in `<directory>`
- `os.getcwd()` returns the path name of the current working directory
- `os.chdir(<path name>)` changes the current working directory to the `<path name>`.

Using the `os` Module

to scan files

All Python scripts are in some directory on disk.

With the `os` module we write utilities
which are independent of the operating system.

Commands we will use:

- `os.listdir(<directory>)` returns a list of strings of names of files and directories in `<directory>`
- `os.getcwd()` returns the path name of the current working directory
- `os.chdir(<path name>)` changes the current working directory to the `<path name>`.

Working with MySQL

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

**scanning files and
grabbing the headers**

filling the table in the
database

1 MySQL basics

starting and stopping the daemon
running the monitor mysql

2 Database for Python scripts

problem statement
create database and table

3 Using MySQLdb

inserting records with Python script
scanning files and grabbing the headers
filling the table in the database

Grabbing the Headers

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
**scanning files and
grabbing the headers**
filling the table in the
database

Every script is documented in a uniform manner:

```
# L-25 MCS 275 Wed 10 Mar 2010 : grabpyhead.py

# Grabs the header line of all .py programs.
# When the file starts with '#!', the header
# line is not the first but the second line.
# Every file has four fields:
# type, number, date, and file name.
```

Format conversion:

```
('L', '25', 'Wed 10 Mar 2010', 'grabpyhead.py')
```

10 Mar 2010

Grabbing the Headers

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
**scanning files and
grabbing the headers**
filling the table in the
database

Every script is documented in a uniform manner:

```
# L-25 MCS 275 Wed 10 Mar 2010 : grabpyhead.py

# Grabs the header line of all .py programs.
# When the file starts with '#!', the header
# line is not the first but the second line.
# Every file has four fields:
# type, number, date, and file name.
```

Format conversion:

```
('L', '25', 'Wed 10 Mar 2010', 'grabpyhead.py')
```

Select only the .py Files

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
**scanning files and
grabbing the headers**
filling the table in the
database

```
import os
```

```
def HasPyExt(name):  
    """  
    Returns True if the name ends in ".py",  
    returns False otherwise.  
    """  
    try:  
        if name[-3:] == '.py':  
            return True  
        else:  
            return False  
    except:  
        return False
```

Select only the .py Files

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
**scanning files and
grabbing the headers**
filling the table in the
database

```
import os
```

```
def HasPyExt(name):
```

```
    """
```

```
    Returns True if the name ends in ".py",  
    returns False otherwise.
```

```
    """
```

```
    try:
```

```
        if name[-3:] == '.py':
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    except:
```

```
        return False
```

Select only the .py Files

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
**scanning files and
grabbing the headers**
filling the table in the
database

```
import os
```

```
def HasPyExt(name):
```

```
    """
```

```
    Returns True if the name ends in ".py",  
    returns False otherwise.
```

```
    """
```

```
    try:
```

```
        if name[-3:] == '.py':
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    except:
```

```
        return False
```

Recognize #! Lines

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def StartPath(line):  
    """  
    Returns True if the line appears to be  
    the path of the python interpreter,  
    returns False otherwise.  
    """  
    try:  
        if line[0:2] == '#!':  
            return True  
        else:  
            return False  
    except:  
        return False
```

Recognize #! Lines

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
**scanning files and
grabbing the headers**
filling the table in the
database

```
def StartPath(line):  
    """  
    Returns True if the line appears to be  
    the path of the python interpreter,  
    returns False otherwise.  
    """  
    try:  
        if line[0:2] == '#!':  
            return True  
        else:  
            return False  
    except:  
        return False
```

Recognize #! Lines

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def StartPath(line):  
    """  
    Returns True if the line appears to be  
    the path of the python interpreter,  
    returns False otherwise.  
    """  
    try:  
        if line[0:2] == '#!':  
            return True  
        else:  
            return False  
    except:  
        return False
```

Split Lines into Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def SplitFields(line):  
    """  
    Returns a tuple with the fields of the header  
    """  
    L0 = line.split(' : '  
    fname = L0[1]  
    L1 = L0[0].split(' MCS 275 '  
    fdate = L1[1]  
    L2 = L1[0].split(' '  
    L3 = L2[1].split('-'  
    ftype = L3[0]  
    fnumb = L3[1]  
    return (ftype, fnumb, fdate, fname)
```

Split Lines into Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def SplitFields(line):  
    """  
    Returns a tuple with the fields of the header  
    """  
    L0 = line.split(' : '  
    fname = L0[1]  
    L1 = L0[0].split(' MCS 275 '  
    fdate = L1[1]  
    L2 = L1[0].split(' '  
    L3 = L2[1].split('-'  
    ftype = L3[0]  
    fnumb = L3[1]  
    return (ftype, fnumb, fdate, fname)
```

Split Lines into Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def SplitFields(line):  
    """  
    Returns a tuple with the fields of the header  
    """  
    L0 = line.split(' : '  
    fname = L0[1]  
    L1 = L0[0].split(' MCS 275 '  
    fdate = L1[1]  
    L2 = L1[0].split(' '  
    L3 = L2[1].split('-'  
    ftype = L3[0]  
    fnumb = L3[1]  
    return (ftype, fnumb, fdate, fname)
```

Split Lines into Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def SplitFields(line):  
    """  
    Returns a tuple with the fields of the header  
    """  
    L0 = line.split(' : '  
    fname = L0[1]  
    L1 = L0[0].split(' MCS 275 '  
    fdate = L1[1]  
    L2 = L1[0].split(' '  
    L3 = L2[1].split('-'  
    ftype = L3[0]  
    fnumb = L3[1]  
    return (ftype, fnumb, fdate, fname)
```

Split Lines into Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def SplitFields(line):  
    """  
    Returns a tuple with the fields of the header  
    """  
    L0 = line.split(' : '  
    fname = L0[1]  
    L1 = L0[0].split(' MCS 275 '  
    fdate = L1[1]  
    L2 = L1[0].split(' '  
    L3 = L2[1].split('-'  
    ftype = L3[0]  
    fnumb = L3[1]  
    return (ftype, fnumb, fdate, fname)
```

Enumerate all Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def EnumFields(d,f):
```

```
    """
```

```
    Enumerates all fields in the header of  
    the .py files in the directory d.  
    For each field, the function f is called.  
    Returns the number of .py files.
```

```
    """
```

```
    L = os.listdir(d)
```

```
    cnt = 0
```

```
    for filename in L:
```

```
        if HasPyExt(filename):
```

```
            cnt = cnt + 1
```

```
            file = open(filename,'r')
```

```
            s = file.readline()
```

```
            if StartPath(s): s = file.readline()
```

```
            f(SplitFields(s[:-1])) # omit \n
```

```
            file.close()
```

```
    return cnt
```

10 Mar 2010

Enumerate all Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def EnumFields(d,f):
    """
    Enumerates all fields in the header of
    the .py files in the directory d.
    For each field, the function f is called.
    Returns the number of .py files.
    """
    L = os.listdir(d)
    cnt = 0
    for filename in L:
        if HasPyExt(filename):
            cnt = cnt + 1
            file = open(filename,'r')
            s = file.readline()
            if StartPath(s): s = file.readline()
            f(SplitFields(s[:-1])) # omit \n
            file.close()
    return cnt
```

10 Mar 2010

Enumerate all Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def EnumFields(d,f):
    """
    Enumerates all fields in the header of
    the .py files in the directory d.
    For each field, the function f is called.
    Returns the number of .py files.
    """
    L = os.listdir(d)
    cnt = 0
    for filename in L:
        if HasPyExt(filename):
            cnt = cnt + 1
            file = open(filename,'r')
            s = file.readline()
            if StartPath(s): s = file.readline()
            f(SplitFields(s[:-1])) # omit \n
            file.close()
    return cnt
```

Enumerate all Fields

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
def EnumFields(d,f):
    """
    Enumerates all fields in the header of
    the .py files in the directory d.
    For each field, the function f is called.
    Returns the number of .py files.
    """
    L = os.listdir(d)
    cnt = 0
    for filename in L:
        if HasPyExt(filename):
            cnt = cnt + 1
            file = open(filename,'r')
            s = file.readline()
            if StartPath(s): s = file.readline()
            f(SplitFields(s[:-1])) # omit \n
            file.close()
    return cnt
```

Callback Functions in Iterators

The `EnumFields` routine is an *iterator*.

An iterator enumerates all items in a collection.

Actions on the items are of no concern of the iterator, display items, select items, or ...

Good software design:

- 1 write iterator separately with simple callback function (e.g.: `print`) to test
- 2 make callback functions specific for applications

Callback Functions in Iterators

The `EnumFields` routine is an *iterator*.

An iterator enumerates all items in a collection.

Actions on the items are of no concern of the iterator,
display items, select items, or ...

Good software design:

- 1 write iterator separately with simple callback function
(e.g.: `print`) to test
- 2 make callback functions specific for applications

Callback Functions in Iterators

The `EnumFields` routine is an *iterator*.

An iterator enumerates all items in a collection.

Actions on the items are of no concern of the iterator,
display items, select items, or ...

Good software design:

- 1 write iterator separately with simple callback function
(e.g.: `print`) to test
- 2 make callback functions specific for applications

Callback Functions in Iterators

The `EnumFields` routine is an *iterator*.

An iterator enumerates all items in a collection.

Actions on the items are of no concern of the iterator, display items, select items, or ...

Good software design:

- 1 write iterator separately with simple callback function (e.g.: `print`) to test
- 2 make callback functions specific for applications

Test Callback and main()

```
def PrintFields(s):  
    """  
    Use as argument to test the enumerator.  
    """  
    print s  
  
def main():  
    """  
    Prints the header of all .py files  
    in the current directory.  
    """  
    n = EnumFields('.', PrintFields)  
    print 'counted %d .py files' % n  
  
if __name__ == "__main__": main()
```

Test Callback and main()

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
**scanning files and
grabbing the headers**
filling the table in the
database

```
def PrintFields(s):  
    """  
    Use as argument to test the enumerator.  
    """  
    print s  
  
def main():  
    """  
    Prints the header of all .py files  
    in the current directory.  
    """  
    n = EnumFields('.',PrintFields)  
    print 'counted %d .py files' % n  
  
if __name__ == "__main__": main()
```

Working with MySQL

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

**filling the table in the
database**

1 MySQL basics

starting and stopping the daemon
running the monitor mysql

2 Database for Python scripts

problem statement
create database and table

3 Using MySQLdb

inserting records with Python script
scanning files and grabbing the headers
filling the table in the database

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for
Python scripts

problem statement
create database and
table

Using
MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

Using the MySQLdb API

Five things to remember:

- 1 **import MySQLdb**
- 2 connect to the database:

```
<connection> =
MySQLdb.connect ( db= " <dbname> " )
```
- 3 create a cursor object:

```
<cursor> = <connection>.cursor ( )
```
- 4 execute mysql commands:

```
<cursor>.execute (<command string> )
```

returns number of rows in the result
- 5 retrieving results of queries:

```
<cursor>.fetchone ( )
```

 returns single row

```
<cursor>.fetchall ( )
```

 returns all rows

```
<cursor>.rowcount
```

 returns number of rows

starting and stopping
the daemon
running the monitor
mysql

problem statement
create database and
table

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

Using the MySQLdb API

Five things to remember:

1 **import MySQLdb**

2 connect to the database:

```
<connection> =  
MySQLdb.connect( db= " <dbname> " )
```

3 create a cursor object:

```
<cursor> = <connection>.cursor()
```

4 execute mysql commands:

```
<cursor>.execute(<command string>)  
returns number of rows in the result
```

5 retrieving results of queries:

```
<cursor>.fetchone() returns single row  
<cursor>.fetchall() returns all rows  
<cursor>.rowcount returns number of rows
```

Using the MySQLdb API

Five things to remember:

1 **import MySQLdb**

2 connect to the database:

```
<connection> =
```

```
MySQLdb.connect( db= " <dbname> " )
```

3 create a cursor object:

```
<cursor> = <connection>.cursor()
```

4 execute mysql commands:

```
<cursor>.execute(<command string>)
```

returns number of rows in the result

5 retrieving results of queries:

```
<cursor>.fetchone() returns single row
```

```
<cursor>.fetchall() returns all rows
```

```
<cursor>.rowcount returns number of rows
```

Using the MySQLdb API

Five things to remember:

1 **import MySQLdb**

2 connect to the database:

```
<connection> =
```

```
MySQLdb.connect( db= " <dbname> " )
```

3 create a cursor object:

```
<cursor> = <connection>.cursor( )
```

4 execute mysql commands:

```
<cursor>.execute(<command string>)
```

returns number of rows in the result

5 retrieving results of queries:

```
<cursor>.fetchone() returns single row
```

```
<cursor>.fetchall() returns all rows
```

```
<cursor>.rowcount returns number of rows
```

Using the MySQLdb API

Five things to remember:

1 **import MySQLdb**

2 connect to the database:

```
<connection> =
```

```
MySQLdb.connect( db= " <dbname> " )
```

3 create a cursor object:

```
<cursor> = <connection>.cursor()
```

4 execute mysql commands:

```
<cursor>.execute(<command string>)
```

returns number of rows in the result

5 retrieving results of queries:

```
<cursor>.fetchone() returns single row
```

```
<cursor>.fetchall() returns all rows
```

```
<cursor>.rowcount returns number of rows
```

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

Using the MySQLdb API

Five things to remember:

1 **import MySQLdb**

2 connect to the database:

```
<connection> =
```

```
MySQLdb.connect( db= " <dbname> " )
```

3 create a cursor object:

```
<cursor> = <connection>.cursor()
```

4 execute mysql commands:

```
<cursor>.execute(<command string>)
```

returns number of rows in the result

5 retrieving results of queries:

```
<cursor>.fetchone() returns single row
```

```
<cursor>.fetchall() returns all rows
```

```
<cursor>.rowcount returns number of rows
```

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

Converting Date Formats

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

```
import os, MySQLdb
from grabpyhead import EnumFields

def MyDate(s):
    """
    Converts a string like Wed 10 Mar 2010
    into the format 2010-03-10.
    """
    m = { "Jan": "01", "Feb": "02", "Mar": "03", \
          "Apr": "04", "May": "05", "Jun": "06", \
          "Jul": "07", "Aug": "08", "Sep": "09", \
          "Oct": "10", "Nov": "11", "Dec": "12" }

    L = s.split(' ')
    day = '%02d' % int(L[1])
    return L[3] + '-' + m[L[2]] + '-' + day
```

Converting Date Formats

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

```
import os, MySQLdb
from grabpyhead import EnumFields

def MyDate(s):
    """
    Converts a string like Wed 10 Mar 2010
    into the format 2010-03-10.
    """
    m = { "Jan": "01", "Feb": "02", "Mar": "03", \
          "Apr": "04", "May": "05", "Jun": "06", \
          "Jul": "07", "Aug": "08", "Sep": "09", \
          "Oct": "10", "Nov": "11", "Dec": "12" }

    L = s.split(' ')
    day = '%02d' % int(L[1])
    return L[3] + '-' + m[L[2]] + '-' + day
```

Converting Date Formats

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

```
import os, MySQLdb
from grabpyhead import EnumFields

def MyDate(s):
    """
    Converts a string like Wed 10 Mar 2010
    into the format 2010-03-10.
    """
    m = { "Jan": "01", "Feb": "02", "Mar": "03", \
          "Apr": "04", "May": "05", "Jun": "06", \
          "Jul": "07", "Aug": "08", "Sep": "09", \
          "Oct": "10", "Nov": "11", "Dec": "12" }
    L = s.split(' ')
    day = '%02d' % int(L[1])
    return L[3] + '-' + m[L[2]] + '-' + day
```

Converting Date Formats

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

```
import os, MySQLdb
from grabpyhead import EnumFields

def MyDate(s):
    """
    Converts a string like Wed 10 Mar 2010
    into the format 2010-03-10.
    """
    m = { "Jan": "01", "Feb": "02", "Mar": "03", \
          "Apr": "04", "May": "05", "Jun": "06", \
          "Jul": "07", "Aug": "08", "Sep": "09", \
          "Oct": "10", "Nov": "11", "Dec": "12" }
    L = s.split(' ')
    day = '%02d' % int(L[1])
    return L[3] + '-' + m[L[2]] + '-' + day
```

MySQL basics

starting and stopping
the daemon

running the monitor
mysql

Database for
Python scripts

problem statement
create database and
table

Using
MySQLdb

inserting records with
Python script

scanning files and
grabbing the headers

filling the table in the
database

Inserting Data

```
def InsertData(c):
    """
    Data is inserted into the database,
    using the cursor c.
    """
    def Insert(s):
        """
        Uses the tuple s to insert into
        the table scripts.
        """
        d = 'insert into scripts values (' \
            + '\"' + s[0] + '\"' + ',' \
            + '\"' + s[1] + '\"' + ',' \
            + '\"' + MyDate(s[2]) + '\"' + ',' \
            + '\"' + s[3] + '\"' + ');'
        c.execute(d)

    n = EnumFields('.',Insert)
    return n
```

starting and stopping
the daemon
running the monitor
mysql

problem statement
create database and
table

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

Inserting Data

```
def InsertData(c):
```

```
    """
```

```
    Data is inserted into the database,  
    using the cursor c.
```

```
    """
```

```
    def Insert(s):
```

```
        """
```

```
        Uses the tuple s to insert into  
        the table scripts.
```

```
        """
```

```
        d = 'insert into scripts values (' \
            + '\"' + s[0] + '\"' + ',' \
            + '\"' + s[1] + '\"' + ',' \
            + '\"' + MyDate(s[2]) + '\"' + ',' \
            + '\"' + s[3] + '\"' + ');'
```

```
        c.execute(d)
```

```
    n = EnumFields('.',Insert)
```

```
    return n
```

starting and stopping
the daemon
running the monitor
mysql

problem statement
create database and
table

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

Inserting Data

```
def InsertData(c):
    """
    Data is inserted into the database,
    using the cursor c.
    """
    def Insert(s):
        """
        Uses the tuple s to insert into
        the table scripts.
        """
        d = 'insert into scripts values (' \
            + '\"' + s[0] + '\"' + ',' \
            + '\"' + s[1] + '\"' + ',' \
            + '\"' + MyDate(s[2]) + '\"' + ',' \
            + '\"' + s[3] + '\"' + ');'
        c.execute(d)

    n = EnumFields('.',Insert)
    return n
```

starting and stopping
the daemon
running the monitor
mysql

problem statement
create database and
table

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

Inserting Data

```
def InsertData(c):
    """
    Data is inserted into the database,
    using the cursor c.
    """
    def Insert(s):
        """
        Uses the tuple s to insert into
        the table scripts.
        """
        d = 'insert into scripts values (' \
            + '\"' + s[0] + '\"' + ',' \
            + '\"' + s[1] + '\"' + ',' \
            + '\"' + MyDate(s[2]) + '\"' + ',' \
            + '\"' + s[3] + '\"' + ');'
        c.execute(d)

    n = EnumFields('.',Insert)
    return n
```

Filling the Table

filldb.py

```
def main():  
    """  
    Prints the header of all .py files  
    in the current directory.  
    """  
    p = os.getcwd()  
    os.chdir('../MCS275py')  
    db = MySQLdb.connect(db="OurPyFiles")  
    c = db.cursor()  
    n = InsertData(c)  
    print 'inserted %d .py files' % n  
    os.chdir(p)
```

Filling the Table

filldb.py

```
def main():
    """
    Prints the header of all .py files
    in the current directory.
    """
    p = os.getcwd()
    os.chdir('../MCS275py')
    db = MySQLdb.connect(db="OurPyFiles")
    c = db.cursor()
    n = InsertData(c)
    print 'inserted %d .py files' % n
    os.chdir(p)
```

Filling the Table

filldb.py

```
def main():  
    """  
    Prints the header of all .py files  
    in the current directory.  
    """  
    p = os.getcwd()  
    os.chdir('../MCS275py')  
    db = MySQLdb.connect(db="OurPyFiles")  
    c = db.cursor()  
    n = InsertData(c)  
    print 'inserted %d .py files' % n  
    os.chdir(p)
```

Filling the Table

filldb.py

```
def main():  
    """  
    Prints the header of all .py files  
    in the current directory.  
    """  
    p = os.getcwd()  
    os.chdir('../MCS275py')  
    db = MySQLdb.connect(db="OurPyFiles")  
    c = db.cursor()  
    n = InsertData(c)  
    print 'inserted %d .py files' % n  
    os.chdir(p)
```

10 Mar 2010

MySQL basics

starting and stopping
the daemonrunning the monitor
mysqlDatabase for
Python scriptsproblem statement
create database and
tableUsing
MySQLdbinserting records with
Python scriptscanning files and
grabbing the headersfilling the table in the
database

Some Queries

Sort by type:

```
mysql> select * from scripts order by d;
```

Recall that `*` is a wild card,
the returned table contains all fields of `scripts`.

To retrieve date and file name of all quiz scripts:

```
mysql> select d, f from scripts where t = "Q";
```

10 Mar 2010

MySQL basics

starting and stopping
the daemonrunning the monitor
mysqlDatabase for
Python scriptsproblem statement
create database and
tableUsing
MySQLdbinserting records with
Python scriptscanning files and
grabbing the headersfilling the table in the
database

Some Queries

Sort by type:

```
mysql> select * from scripts order by d;
```

Recall that `*` is a wild card,
the returned table contains all fields of `scripts`.

To retrieve date and file name of all quiz scripts:

```
mysql> select d, f from scripts where t = "Q";
```

10 Mar 2010

MySQL basics

starting and stopping
the daemonrunning the monitor
mysqlDatabase for
Python scriptsproblem statement
create database and
tableUsing
MySQLdbinserting records with
Python scriptscanning files and
grabbing the headersfilling the table in the
database

Some Queries

Sort by type:

```
mysql> select * from scripts order by d;
```

Recall that `*` is a wild card,
the returned table contains all fields of `scripts`.

To retrieve date and file name of all quiz scripts:

```
mysql> select d, f from scripts where t = "Q";
```

A Script to view all Rows

viewdbdata.py

```
import MySQLdb

def main():
    """
    Executes a simple query to the database.
    """
    db = MySQLdb.connect(db="OurPyFiles")
    c = db.cursor()
    q = 'select * from scripts order by d'
    lc = c.execute(q)
    print 'found %d rows' % int(lc)
    while True:
        print c.fetchone()
        ans = raw_input('see more ? (y/n) ')
        if ans != 'y': break
```

A Script to view all Rows

viewdbdata.py

```
import MySQLdb

def main():
    """
    Executes a simple query to the database.
    """
    db = MySQLdb.connect(db="OurPyFiles")
    c = db.cursor()
    q = 'select * from scripts order by d'
    lc = c.execute(q)
    print 'found %d rows' % int(lc)
    while True:
        print c.fetchone()
        ans = raw_input('see more ? (y/n) ')
        if ans != 'y': break
```

A Script to view all Rows

viewdbdata.py

```
import MySQLdb

def main():
    """
    Executes a simple query to the database.
    """
    db = MySQLdb.connect(db="OurPyFiles")
    c = db.cursor()
    q = 'select * from scripts order by d'
    lc = c.execute(q)
    print 'found %d rows' % int(lc)
    while True:
        print c.fetchone()
        ans = raw_input('see more ? (y/n) ')
        if ans != 'y': break
```

10 Mar 2010

Summary and Exercises

MySQL basics

starting and stopping
the daemon
running the monitor
mysql

Database for Python scripts

problem statement
create database and
table

Using MySQLdb

inserting records with
Python script
scanning files and
grabbing the headers
filling the table in the
database

More of Chapter 11 of *Making Use of Python*,
see also <http://www.python.org/doc/topics>.

- 1 Extend the script `filldb.py` so that it also recursively looks for `.py` files in all subdirectories.
- 2 The database is not normalized because type, number, and dates are redundant. Use `mysql` to select from scripts to create a table `typedates` to store only data like "L-25" and "10 Mar 2010". Select from scripts to create a table `typefiles` to store "L-25" and "viewdbdata.py".
- 3 Modify the Python script `filldb.py` so it fills the tables `typedates` and `typefiles` (defined in the previous exercise) while scanning the `.py` files.
- 4 Extend the script `viewdbdata.py` asking for sorting directives. There are 32 possible orders.