Introduction to Programming with Python

II.5. Data persistence

Oscar Delgado
oscar.delgado@uam.es

Data persistence

Data persistence

Files	Objects serialization	Relational databases
• Text, binary formats • Sequencial read • Suitable for small, mediuam datasets (<2,3GB)	• Useful for storing “full” objects (value and structure) • Simple and easy • Only suitable for small objects	• More complex to setup and to access • In theory, no limit in the dataset size

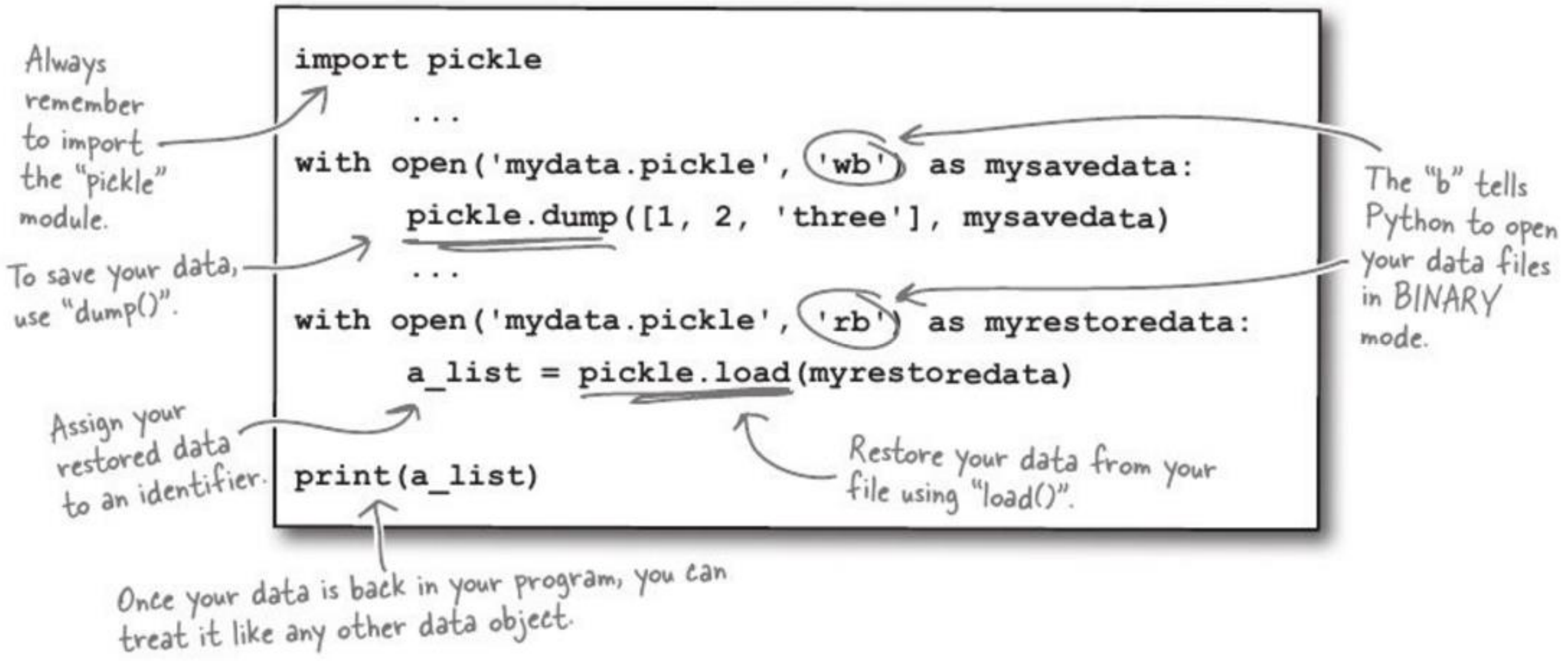
Object serialization

Object persistence

Object serialization Useful to store a full object in a file: both structure and values.

- Any object (lists, dictionaries, custom objects) can be serialized.
- The format is specific to Python (even to a specific Python version)

Pickling objects



```
import pickle
        ...
with open('mydata.pickle', 'wb') as mysavedata:
        pickle.dump([1, 2, 'three'], mysavedata)
        ...
with open('mydata.pickle', 'rb') as myrestoredata:
        a_list = pickle.load(myrestoredata)

print(a_list)
```

Always remember to import the "pickle" module.
To save your data, use "dump()".
Assign your restored data to an identifier.
The "b" tells Python to open your data files in BINARY mode.
Restore your data from your file using "load()".
Once your data is back in your program, you can treat it like any other data object.

Relational Databases

Connection

```
import MySQLdb

# Open database connection
db = MySQLdb.connect(<<SERVIDOR>>,<<USUARIO>>,<<CONTRASEÑA>>,<<BASE DE DATOS>> )

# disconnect from server
db.close()
```

INSERT operations

```
import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.
sql = "INSERT INTO EMPLOYEE(FIRST_NAME, \
      LAST_NAME, AGE, SEX, INCOME) \
      VALUES ('%s', '%s', '%d', '%c', '%d' )" % \
      ('Mac', 'Mohan', 20, 'M', 2000)
```

```
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()

# disconnect from server
db.close()
```

Read operations

Querying database Once our database connection is established, you are ready to make a query into this database:

- **fetchone():** It fetches the next row of a query result set.
- **fetchall():** It fetches all the rows in a result set
- **rowcount:** This is a read-only attribute and returns the number of rows that were affected by an execute() method.

Read operations

```
import MySQLdb

# Open database connection
db =
MySQLdb.connect("localhost","testuser","test123","T
ESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

sql = "SELECT * FROM EMPLOYEE \
      WHERE INCOME > '%d'" % (1000)
try:
  # Execute the SQL command
  cursor.execute(sql)
# Fetch all the rows in a list of lists.
  results = cursor.fetchall()
  for row in results:
    fname = row[0]
    lname = row[1]
    age = row[2]
    sex = row[3]
    income = row[4]
    # Now print fetched result
    print "fname=%s,lname=%s,age=%d,sex=%s,income=%d" % \
         (fname, lname, age, sex, income )
except:
  print "Error: unable to fecth data"

# disconnect from server
db.close()
```

UPDATE operations

```
import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to UPDATE required records
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1
                          WHERE SEX = '%c'" % ('M')
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()

# disconnect from server
db.close()
```

DELETE operations

```
import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","test user","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()

# disconnect from server
db.close()
```

Escuela Politécnica Superior

UNIVERSIDAD AUTONOMA DE MADRID