Introduction to Databases and Data Management

Embedded Interface Design

with **Bruce Montgomery**

Learning Objectives

- Students will be able to…
  - Consider and apply various ways to maintain data in their programs
  - Recognize the differences between SQL and non-SQL databases

Maintaining Data

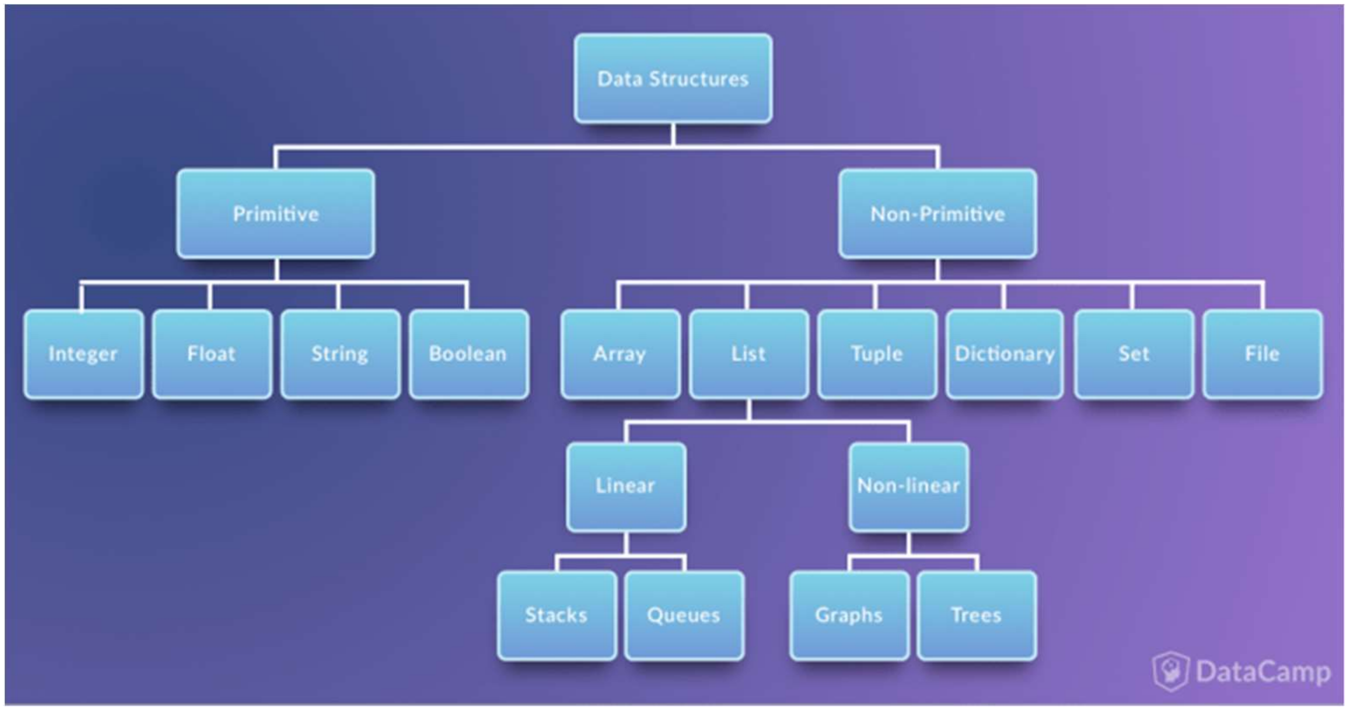
- For any system, embedded or not, some data must be maintained in volatile storage (will only exist until freed or power is removed) and non-volatile (maintained even after power is removed)
- Typically, volatile memory is in RAM (Random Access Memory), and often data is loaded into RAM on startup from other sources
- Non-volatile data storage could be in flash, power-protected RAM, hard drives, and solid state storage, or connected systems in the cloud, for example

Data management for prototype systems

- Rather than model a detailed embedded storage system, in a prototype, data is often stored in the easiest way possible
  - This may be in-memory data structures; arrays and dictionaries or other structures
  - This may be in local embedded databases
  - Or could be in external databases on other systems or in the cloud
- The issues for prototype storage will include speed of access, amounts and types of data, and the uses for the data, to name a few considerations

Data management in Python

- Data in Python natively uses primitive and non-primitive structures
- Primitive data storage essentially uses single simple variables
- Non-primitive storage are various more complex multi-value structures
- All such structures are volatile unless specifically saved to files or set as constants



Native Python Data Structures [1]

Relational Databases

- A database is a utility for saving persistent data and for searching and relating various data elements and sources [2]
- The most common databases are relational databases
  - Relational databases store data in tables, and use key references to relate how one data row in a table might relate to another data row in another table
  - Data tables are created using a definition language, often SQL (Structured Query Language), and other SQL elements are used to create, read, update, and delete data (the so-called CRUD operations)
  - Data in relational databases can be worked with using a language like SQL, or with an intermediate tool called an Object-Relational Mapper (ORM), which uses native code like Python to provide data access
  - SQL is fairly standard across database applications, ORMs are often more unique

NoSQL Databases

- There are a number of data storage tools that do not use SQL, and are designed for specific performance or usage profiles [3]
- Typical NoSQL data stores include
  - Key-value pairs – examples: Redis, Memcached
  - Document-oriented – example: Mongo-DB
  - Column-family tables – example: Cassandra
  - Graphs – examples: Neo4j, Cayley, Titan
- Redis, for instance, is often used for web applications where fast response time for session data is required

Common Database Tools for Python

- Relational Databases
  - SQLite – built into Python, limited to single connections
  - MySQL – easiest to pick up and use
  - PostgreSQL – most feature rich of the set
- NoSQL Databases
  - Redis – Go to tool for most data caching speed-dependent applications
  - MongoDB – stores JSON documents for more complex data
- AWS Data Storage (more about Cloud tools soon)
  - AWS Elasticache offers Redis and Memcached implementations
  - AWS Aurora – MySQL, PostgreSQL compatible
  - AWS RDS (Relational Database Service) – six common relational database engines

Typical MySQL interaction with Python

- This example is from an online MySQL tutorial [4]
- Steps include
  - Install MySQL
  - Database setup
  - Python data access

```
sudo apt-get install python-mysqldb
```

```
mysql -u USERNAME –p
mysql&gt; CREATE DATABASE pythonspot;
mysql&gt; USE pythonspot;
CREATE TABLE IF NOT EXISTS examples (
  id int(11) NOT NULL AUTO_INCREMENT,
  description varchar(45),
  PRIMARY KEY (id)
);
INSERT INTO examples(description) VALUES ("Hello World");
INSERT INTO examples(description) VALUES ("MySQL Example");
INSERT INTO examples(description) VALUES ("Flask Example");
```

Typical MySQL interaction with Python

- This example is from an online MySQL tutorial [4]
- Steps include
  - Install MySQL
  - Database setup
  - Python data access

```
#!/usr/bin/python3
import MySQLdb

db = MySQLdb.connect(host="localhost",  # your host
                     user="root",       # username
                     passwd="root",     # password
                     db="pythonspot")   # name of the database

# Create a Cursor object to execute queries
cur = db.cursor()
# Select data from table using SQL query.
cur.execute("SELECT * FROM examples")
# print the first and second columns
for row in cur.fetchall() :
   print row[0], " ", row[1]
```

Typical Redis interaction with Python

- This is a “Hello, World” example from an online Redis tutorial [5]
- Steps include
  - Install Redis
  - Install redis-py
  - Python data access
- More information at redis-py site [6]

```
sudo apt-get install redis-server
```

```
pip3 install redis-py
```

```
#!/usr/bin/python3
import redis

# connect to your redis instance from redis-py using defaults
r = redis.Redis( host=‘localhost’, port=6379, db=0)

# write to redis using key “greet”, value “Hello World!”
r.set(“greet”,”Hello World!”)

# read from redis using the key “greet”
value = r.get(“greet”)
print(value)        # Outputs “Hello World!”
```

Next Steps

- Hardware Handout today – tracking sheet (view only) at
  - https://docs.google.com/spreadsheets/d/1mKPs_GmbZGsNRROy2-BztipEiR1NFhsgTi9uQJ-Lerk/edit?usp=sharing
  - Shipped to distance students Saturday, see your e-mail
  - You should have a RPi3, a power supply, an SD card, and a DHT22 sensor
    - You will source your own USB cables and resistor/wiring for DHT22
- Review Project 1 – submission links are up on Canvas
- Background survey is up – please complete by Sunday 9/15 – participation grade element
- Wednesday: M2M, Low Level Protocols, IoT Protocols
- Next week: M2M Protocols, LPWAN Protocols, Cloud Architectures, Cloud for IoT, AWS, AWS for IoT

References

[1] https://www.datacamp.com/community/tutorials/data-structures-python
[2] https://www.fullstackpython.com/databases.html
[3] https://www.fullstackpython.com/no-sql-datastore.html
[4] https://pythonspot.com/mysql-with-python/
[5] https://redislabs.com/lp/python-redis/
[6] https://github.com/andymccurdy/redis-py/