Computer Science & Engineering 120
Learning to Code

Introduction to Data

Christopher M. Bourke
cbourke@cse.unl.edu

Part I: Working With Data

Topic Overview

Data

- Data Formats
- Data Operations

Introduction

- Data models real-world problems
- Data is used to make decisions
- Data provides information
- Understanding data gives insights, knowledge

Data as a Table

First Name	Last Name	NUID	Email	Year	GPA	Course Number	Course Name
Starlin	Castro	12301013		Sophomore	3.75	CSCE 120	Learning To Code
Starlin	Castro	12301013		Sophomore	3.75	MATH 103	College Algebra & Trigonometry
Starlin	Castro	12301013		Sophomore	3.75	MUNM 287	History of Rock Music
Anthony	Rizzo	44001244	arizzo@mlb.com	Freshman	3.24	CSCE 120	Learning To Code
Anthony	Rizzo	44001244	arizzo@mlb.com	Freshman	3.24	MUNM 287	History of Rock Music
Anthony	Rizzo	44001244	arizzo@mlb.com	Freshman	3.24	PSYC 181	Introduction to Psychology
Edwin	Jackson	00321023	ejackson@unl.edu	Senior	2.95	MRKT 257	Sales Communication
Edwin	Jackson	00321023	ejackson@unl.edu	Senior	2.95	FINA 260	Personal Finance
Brett	Jackson	93213394	brett.jackson@gmail.com	Freshman	3.8	CSCE 120	Learning To Code
Brett	Jackson	93213394	brett.jackson@gmail.com	Freshman	3.8	BLAW 372	Business Law I
Javier	Baez	33928192	jbaez@cubs.com	Freshman	3.21	ECON 211	Principles of Macroeconomics
Javier	Baez	33928192	jbaez@iacubs.com	Freshman	3.21	BLAW 372	Business Law I
Javier	Baez	33928192	jbaez@iacubs.com	Freshman	3.21	ENGL 150	Writing: Rhetoric as Inquiry
Junior	Lake	11223344	j_lake@yahoo.com	Sophomore	2.81	MUNM 287	History of Rock Music
Junior	Lake	11223344	jlake@gmail.com	Sophomore	2.81	CSCE 120	Learning To Code
Richard	Renteria	89320191	drenteria@gmail.com	Senior	3.91	ENGR 100	Interpersonal Skills for Engineering Leaders
Richard	Renteria	89320191	drenteria@gmail.com	Senior	3.91	CSCE 120	Learning To Code
Richard	Renteria	89320191	rrenteria@cubs.com	Senior	3.91	PHYS 211	General Physics I
Ryne	Sandberg	33221232	sandberg@mlb.com	Junior	3.45	BLAW 300	Business, Government & Society
Ryne	Sandberg	33221232	sandberg@mlb.com	Junior	3.45	CSCE 477	Cryptography & Security

Data as a Table

- Easy to read, but
- Difficult for a computer to process
- Difficult to manipulate
- Static: no real way to discern any patterns or statistics beyond individual records

Data as a Flat File I

```
firstName,lastName,nuid,email,year,gpa,courseNumber,courseName
Starlin,Castro,12301013,scastro@cubs.com,Sophomore,3.75,CSCE 120,Learning To Code
Starlin,Castro,12301013,scastro@cubs.com,Sophomore,3.75,MATH 103,College Algebra & Trigonometry
Starlin,Castro,12301013,scastro@cubs.com,Sophomore,3.75,MUNM 287,History of Rock Music
Anthony,Rizzo,44001244,arizzo@mlb.com,Freshman,3.24,CSCE 120,Learning To Code
Anthony,Rizzo,44001244,arizzo@mlb.com,Freshman,3.24,MUNM 287,History of Rock Music
Anthony,Rizzo,44001244,arizzo@mlb.com,Freshman,3.24,PSYC 181,Introduction to Psychology
Edwin,Jackson,00321023,ejackson@unl.edu,Senior,2.95,MRKT 257,Sales Communication
Edwin,Jackson,00321023,ejackson@unl.edu,Senior,2.95,FINA 260,Personal Finance
Brett,Jackson,93213394,brett.jackson@gmail.com,Freshman,3.8,CSCE 120,Learning To Code
Brett,Jackson,93213394,brett.jackson@gmail.com,Freshman,3.8,BLAW 372,Business Law I
Javier,Baez,33928192,jbaez@cubs.com,Freshman,3.21,ECON 211,Principles of Macroeconomics
Javier,Baez,33928192,jbaez@iacubs.com,Freshman,3.21,BLAW 372,Business Law I
Javier,Baez,33928192,jbaez@iacubs.com,Freshman,3.21,ENGL 150,Writing: Rhetoric as Inquiry
Junior,Lake,11223344,j_lake@yahoo.com,Sophomore,2.81,MUNM 287,History of Rock Music
Junior,Lake,11223344,jlake@gmail.com,Sophomore,2.81,CSCE 120,Learning To Code
Richard,Renteria,89320191,drenteria@gmail.com,Senior,3.91,ENGR 100,Interpersonal Skills for Engineering Leaders
Richard,Renteria,89320191,drenteria@gmail.com,Senior,3.91,CSCE 120,Learning To Code
Richard,Renteria,89320191,rrenteria@cubs.com,Senior,3.91,PHYS 211,General Physics I
Ryne,Sandberg,33221232,sandberg@mlb.com,Junior,3.45,BLAW 300,"Business, Government & Society"
Ryne,Sandberg,33221232,sandberg@mlb.com,Junior,3.45,CSCE 477,Cryptography & Security
```

Data as a Flat File II

- Comma Separated Value File (CSV)
- Columns (fields) and rows (records)
- Not intended for human consumption
- Easier for a computer program to process
- Still lacks sophistication

Data as a Spreadsheet

- Same data can be imported into a spreadsheet such as Excel
- Spreadsheets provide means to manipulate and process data
- Data is tied to a (proprietary) format
- Operations are limited to what functionality the program provides

Extensible Markup Language (XML)

```xml
<?xml version="1.0"?>
<roster>
    <enrollment>
        <firstName>Starlin</firstName>
        <lastName>Castro</lastName>
        <nuid>12301013</nuid>
        <email>scastro@cubs.com</email>
        <year>Sophomore</year>
        <gpa>3.75</gpa>
        <courseNumber>CSCE 120</courseNumber>
        <courseName>Learning To Code</courseName>
    </enrollment>
    ...
    <enrollment>
        <firstName>Ryne</firstName>
        <lastName>Sandberg</lastName>
        <nuid>33221232</nuid>
        <email>sandberg@mlb.com</email>
        <year>Junior</year>
        <gpa>3.45</gpa>
        <courseNumber>CSCE 477</courseNumber>
        <courseName>Cryptography &amp;amp; Security</courseName>
    </enrollment>
</roster>
```

Extensible Markup Language (XML)

- Open, standardized format
- Interoperable
- Each piece of data is "marked up" with a *tag*
- Semantics give meaning to data
- Data has a nested *tree structure*: parent-child relationship

JavaScript Object Notation (JSON)

```
{
  "roster": [
  {
    "firstName":"Starlin",
    "lastName":"Castro",
    "nuid":12301013,
    "email":"scastro@cubs.com",
    "year":"Sophomore",
    "gpa":3.75,
    "courseNumber":"CSCE 120",
    "courseName":"Learning To Code"
  },
  ...
  {
    "firstName":"Ryne",
    "lastName":"Sandberg",
    "nuid":33221232,
    "email":"sandberg@mlb.com",
    "year":"Junior",
    "gpa":3.45,
    "courseNumber":"CSCE 477",
    "courseName":"Cryptography & Security"
  }
  ]
}
```

JavaScript Object Notation (JSON)

- Open, standardized, interoperable
- More light-weight: fewer characters (less storage, quicker transmission time)
- Same *tree* structure
- Data is stored as key-value pairs
- A subset of JavaScript

Data Operations

Data Transformation

- Electronic Data Interchange (EDI)
- Transforming data in one format into another: CSV $\rightarrow$ XML, Excel $\rightarrow$ JSON, etc.
- Facilitates communication between different systems

Data Operations

Data Organization

- Sorting
- Searching
- Efficiency, scalability
- Data *normalization*

Data Operations

Data Aggregation

- Computing statistics: count, average, etc.
- Example: total number of enrolled students, total number of courses, average number of students per course, etc.
- Grouping of data: a total number of credit hours for each student, a total enrollment count for each class, etc.

Data Operations

Data Visualization

- Computers are good at (large amounts of) raw data, humans aren't
- Computers are not good at recognizing patterns, humans are
- Visualizing data can make it clearer to a human user
- Simple: bar graph, pie charts, etc.
- Advanced: heat maps, connection graphs

Data Operations

Data Mining

- Advanced techniques: Data Mining and Machine Learning
- Discover structures and patterns and information that you may not have been looking for
- Make predictions or models
- Clustering & Classification

Part II: JavaScript Object Notation (JSON)

Topic Overview

JavaScript Object Notation

- Overview
- Data Types
- Proper Formatting
- Data Errors

Overview

JavaScript Object Notation

- A subset of JavaScript
- Natively recognized and used directly in JavaScript
- Begins with a *root object* with opening/closing curly brackets
- Each piece of data is represented as a key-value pair:
  `"key": value`
- Key is a *string* denoted with double quotes
- Value can be a number of *types*
- Each key-value pair is separated by a comma

Example

```
{
  "student": {
    "firstName": "John",
    "lastName": "Student",
    "nuid": 12345678,
    "gpa": 3.85,
    "emails": ["jstudent@unl.edu", "johnny@gmail.com"]
  },
  "course": {
    "id": 4231,
    "name": "Learning to Code",
    "code": "CSCE 120"
  }
}
```

More

- Keys must be unique (duplicates are ignored)
- Only strings can be used as keys
- Keys are *case sensitive*: `"name": "Chris"` is not the same as `"Name": "Chris"`
- Best Practice: use `"lowerCamelCasing"` for keys

JSON Data Structures are:

- Nested
- Form a *tree* structure

Data Types

Numeric Types

JavaScript supports *numeric* data types

- Can use integers or decimal numbers
- Typing numbers directly as values are called *literals*
- `"amount": 5` and `"amount": 5.0` are the same
- Computers are finite machines: not all values can be expressed; maximum precision is about 14–16 decimal places

```
{
  "costPerUnit": 123.5,
  "numUnits": 10,
  "pi": 3.14159,
  "milesPerHour": 60
}
```

Data Types

Strings

- A *string* is a collection of ordered characters
- May include any printable character (or Unicode characters for international symbols)
- A string begins and ends with double quotes
- Special characters need to be *escaped* with a backslash: `\", \\, \n, \t`

```
{
  "firstName": "John",
  "nickname": "John \"The Man\" Student",
  "homepage": "http://cse.unl.edu"
}
```

Data Types

Booleans

- A *boolean* is a value that is either *true* or *false*
- Keywords: `true` and `false`
- Not strings, no double quotes used

```
{
  "isFaculty": true,
  "isRegistered": false
}
```

Data Types

Arrays

- An *array* is a collection of *ordered* elements
- Syntax: use square brackets to begin/end
- Each element is a value (no key)
- Each element is separated by a comma

```
{
  "courses": ["MATH106", "CSCE120", "MUNM287"],
  "enrollments": [23, 27, 132]
}
```

Data Types

Objects

- An *object* is a collection of **un***ordered* elements
- An object *encapsulates* a set of key-value data
- Root object
- Nested objects

Data Types

The `null` value

- The `null` value is a special keyword
- Used to indicate missing, unknown, or invalid data
- Examples:

```
{
  "gradCourses": null,
  "emailPreference": null
}
```

JSON Formatting

- All brackets and quotes must be *well-balanced*: closed and properly nested
- No trailing commas
- Compact vs "pretty print" representation:

```
{"student":{"firstName":"John","lastName":"Student","nuid":1
```

- Fewer characters means less storage, quicker transmission time

Data Errors

Formatting Errors

- Invalid formatting means that a program can't parse your data
- Program can't (and *shouldn't*) attempt to "interpret" what you meant

Data Errors

Syntax Errors

- Data may be formatted correctly, but contain syntax errors
- Example: misspelled key value
- Misuse of whitespace inside keys or incorrect casing
- Consistent naming convention (lower camel casing) can prevent this

Data Errors

Consistency Errors

- Data may be well-formatted and free of syntax errors
- May still be “bad” garbage data
- Misspelling of data *values*
- Numeric values that are out of range (negative percentage)
- Missing data values
- Misidentified data items
- Inconsistent data (a name with multiple formats or spellings)

Further exploration in the Hacktivities...