Programming and Database Fundamentals for Data Scientists

Python Lists

Varun Chandola

School of Engineering and Applied Sciences
State University of New York at Buffalo
Buffalo, NY, USA
chandola@buffalo.edu

University at Buffalo
School of Engineering and Applied Sciences

Outline

Recap

Basic Data Types

Lists

What do we know so far?

- Python is awesome!!
- Forced indentations, simultaneous assignments, swapping values in one line
- Read - *The Zen of Python*
- What next?

Basic Data Types

- `float` - real numbers
- `int` - integers
- `str` - string, text
- `bool` - true, false

Basic Data Types

- `float` - real numbers
- `int` - integers
- `str` - string, text
- `bool` - true, false

- Each type refers to one value

Numeric Data Types

- How do I figure out the *type* of a variable?

Numeric Data Types

- How do I figure out the *type* of a variable?
  - Use in-built function `type`
- Why not use `float` instead of using `float` and `int`?

Numeric Data Types

- How do I figure out the *type* of a variable?
  - Use in-built function `type`
- Why not use `float` instead of using `float` and `int`?
  - Most mathematical algorithms are very efficient with integers
- Quick question
  - What is `7/3`?
  - What is `7//3`?
    - Also known as *integer division*.

Type Conversions

- Python supports valid type conversions
- `float(3)`
- `int(‘‘3’’)`
- A good way to validate inputs
- `try-catch` primer

Errors and Exceptions

- Typically two types of errors are encountered: *syntax errors* and *exceptions*
- Syntax errors are reported by the parser and are (relatively) easier to fix
- Exceptions occur during run time
- Unhandled exceptions crash the application
- How to explicitly handle exceptions?
  - `try` and `except` (with optional `else`)
- Cleaning up
  - Use `finally`

Lists

- In Data Science we work with many data points (recall the Chicago Crime example)
- Creating one variable for each data point is inefficient
- Introducing - *Python Lists*

Python Lists

- A collection of values
- Can contain any type
- Can contain several different types

Indexing Lists

- Zero based indexing
- Accessing from the end of the list

Manipulating Lists

- Allows adding, appending, deleting elements

List Storage

- A new list is created everytime we use `[]`
- But what about?
  - `A = list(range(10))`
  - `B = A`

List Storage

- A new list is created everytime we use `[]`
- But what about?
  - `A = list(range(10))`
  - `B = A`
- `A` and `B` *point* to the same collection

Some more operations on lists

- **list**.extend
- **list**.insert
- **list**.remove
- **list**.pop
- **list**.clear
- **list**.index
- **list**.count
- **list**.reverse
- **list**.sort

Lists of Lists

- Nested lists

To Summarize

- The list object stores pointers to objects, not the actual objects themselves.
  - The size of a list in memory depends on the number of objects in the list, not the size of the objects.
- Getting or setting elements is $O(1)$
- Appending is more expensive but not $O(n^2)$, where n is the length of the list
- Removing items is similar in complexity as adding elements
- The time needed to reverse a list is proportional to the list size (O(n)).
- The time needed to sort a list varies; the worst case is O(n log n), but typical cases are often a lot better than that.

References