1. Unit 01 - Foundation and Tidy Data Model

- two-dimensional Python data structures to represent tabular data
  - list of row lists
  - dictionary mapping column names to a list of column values
  - list of row dictionaries, where each dictionary maps a column name to the value for that row
- Regular expressions/pattern matching/extraction
  - given a regular expression, understand what it matches
  - write regular expressions to problem solve
- fundamental operations on dataframes
  - subset rows and columns based on
    - integer indices (individual or slice or list)
    - row or column names (individual or slice or list)
    - boolean predicate
  - aggregate operations
    - across full set (one column, all rows)
    - using group_by partitioning
  - iterating over rows
  - column vector operations and adding columns
  - sorting, ordering (less important for final)
  - do not need to know merge/join here
- know the constraints of the model, to be able to
  - recognize when data is not tidy
    - tidy characteristics
      - observations
        - unique independent variables
        - set of measures/dependent variables *determined by* value-combination of independent variables
      - variables/columns
    - red flags
      - time going across columns
      - new data growing in column direction without being a *new measure*
  - understand what operations can transform two dimensional data into tidy form
    - melt and pivot and what arguments they need to do their job

2. Unit 02 - Relational Data Model

- Sound database design (constraints of the model)
  - recognize the big violations and know what to do about them to achieve a sound database design
    - multiple "things" in one table
    - multipart (in column name)
    - multi-value (as value for a given column)
    - derived data
    - note that solution (multi-value or two-things-in-one) requires a new table or tables, based on whether they are one-many or many-many
  - Schema
    - keys (primary vs foreign, composite vs single-field)
    - relationships
      - one-many
      - linkage table to get many-many from one-many many-one
  - Draw/Design a Schema that is sound
- SQL/Operations
  - SELECT/WHERE

- INNER JOIN, LEFT OUTER JOIN
- Subquery to break a problem into smaller pieces
- GROUP_BY and aggregation in SELECT

> Note that the online part of the exam can use python magics

3. Unit 03 - Networking and HTTP

- HTTP as messages (request/response) over TCP
  - General syntax
  - GET versus POST (know that there are others, but don't need specifics beyond GET/POST)
  - Headers
  - POST body
  - Response
    - status code
    - headers
    - body (as html, or xml, or csv, or binary like jpeg)
- `requests` module to create request and be able to do something useful with response

4. Unit 04 - Tree Data Model - XML, JSON

- As a file format
  - Write well-formed XML
  - Determine if XML is **well-formed**, and therefore parseable
  - tags, attributes, text
- Tree operations
  - Given JSON or XML (and know structure), how to
    - loop over children
    - retrieve tags, attributes, text
  - XPath declarative extraction of data
    - tree tag and attribute traversal
    - conditionals
    - positional
- Constraints in the form of DTD for XML
  - Given DTD and XML, determine if **valid**

5. Unit 05 - APIs, Authentication/Authorization; Synthesis

- Understand how a data provider can offer data to clients over HTTP by specifying
  - protocol and location
  - resource endpoint and path parameters
  - Query parameters
  - request header values
  - POST body (Form) parameters
  - Tie content response to downstream processing as XML/JSON/HTML
- Understand OAuth 2
  - Conceptually why we want delegated authorization and how that differs from impersonation (user/password known/used by App) (i.e the three party system)
  - Steps in the OAuth Dance
    - Lines 1-6 to get authorization code
      - building the url
      - resource owner steps
      - redirect to get code
    - Lines 7-8 to exchange code for token
    - Lines 9-10 for API request/response with token
- Web Scraping

- Basic HTML as a tree instantiating XML concepts
- Common scenarios for HTML to carry tabular data
  - as table/tr/td/th