* “=” is used to assign a value to a variable whereas “==” is to compare two values
* String concatenation: cannot concatenate mixed types

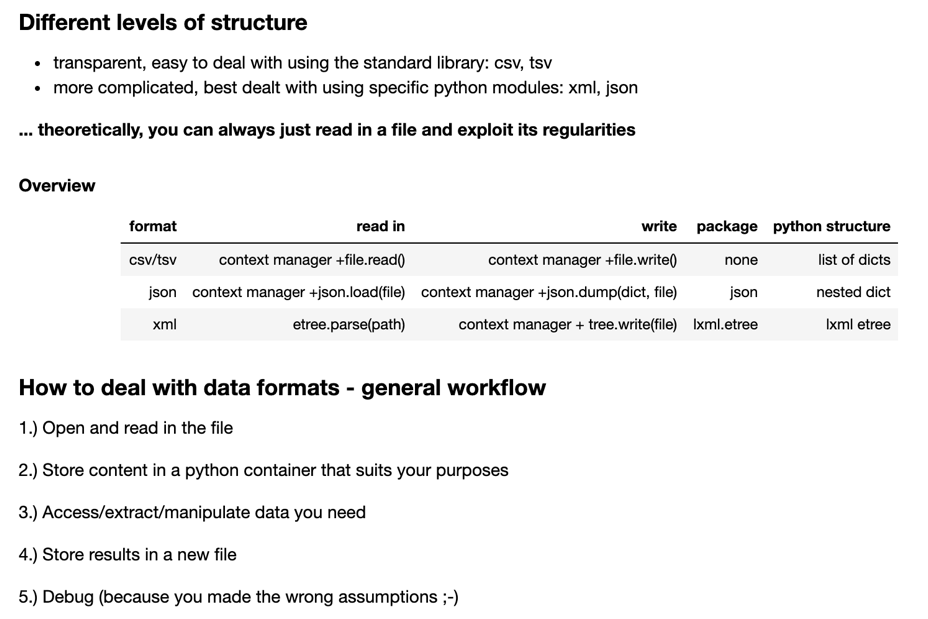
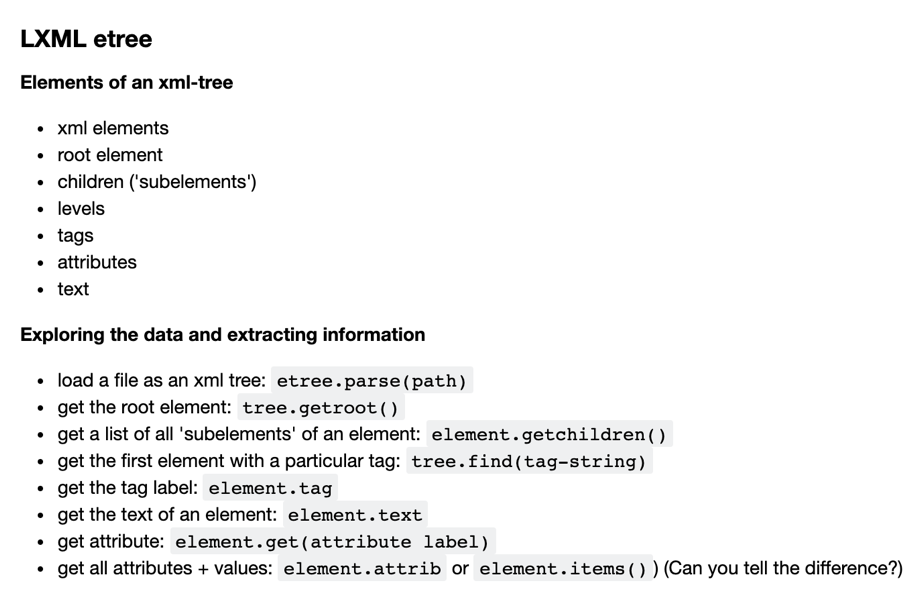
Some Hints:

* String methods
* Loops, functions, scope

1. A function is a reusable block of code that performs a specific task

* Data formats
* Standard lib

Some Notes for previous exams:

* Print( ) function writes, i.e., “prints”, a str or a number on the console. Return( ) statement does not print out the value and it returns when the function is called
* A global variable can be reached anywhere in the code while a local only in the scope
* List all files in a directory: os.walk( ) or glob.glob( )
* No Difference between if not and if a not in
* What is the difference between str() and json.dumps() to convert a Python dictionary to a string
* 
* 
* XML: attribute, elements, items

1. Two types: lxml.etree\_ElementTree (by etree.fromstring) or lxml.etree.\_Element (by etree.parse)
2. e.g. <person role="coordinator">Van der Vliet</person>

* Attributes: role and other info in the line
* Class <Element> : the whole line
* Text: Van der Vliet – element information (str)
* Tag: </person> - element information (str)

Some Notes for Q&A:

* <https://www.programiz.com/python-programming/strings-method>
* Counting number inside a for loop (counter = 0) > count occurrence
* Import a full py file and import a single function

e.g. import nltk > call nltk.word\_tokenize

e.g. From nltk import word\_tokenize

Some Notes for Python:

1. Name of variables:

* Start with a letter or underscore (\_)
* Only contain letters, numbers and underscores
* TypeError

1. Type Casting: str > int, etc.
2. Float: numbers have digits after the dot (23.45)
3. X % Y: remainder of x/y (e.g. 2 % 4 = 2)
4. X // Y: floored quotient of x and y (integer division, 5 // 3 = 1; 5 / 3 = 1.6666667)
5. SyntaxError (e.g. print (x=’windy’s’))

* “windy’s”
* ‘windy\’s’
* “windy\”s”

1. Multiline strings:

* Single or double quotes with (\n\)
* Three single or double quotes
* (\) make no difference to the long strong

1. Repr ( ): internal representation
2. (\t): white space; (\n): new line
3. String slicing:

* My\_string[a:b:c]
* a.starting point
* b.ending before b
* c.stepsize
* my\_string[ : : -1]: reverse the order in the string

1. string is immutable
2. string order: UPPER letter then lower letter

* A, B, C then a, b, c
* Z then a

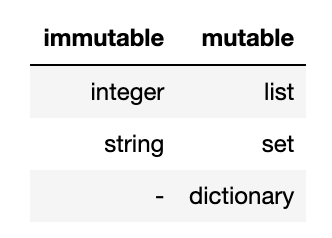
1. String concatenation: consistency (e.g. str + str)
2. F string: f “….. { }”
3. Str.strip: a string without leading or ending whitespace

* String.lstrip( )
* String.rstrip( )

1. Membership operators (in, not in) can only be used in iterables (e.g. string, list; not in integer)

* TypeError

1. If-if statement: code will check all conditions
2. If-elif statement: one true, then no check on the rest
3. Mutability:

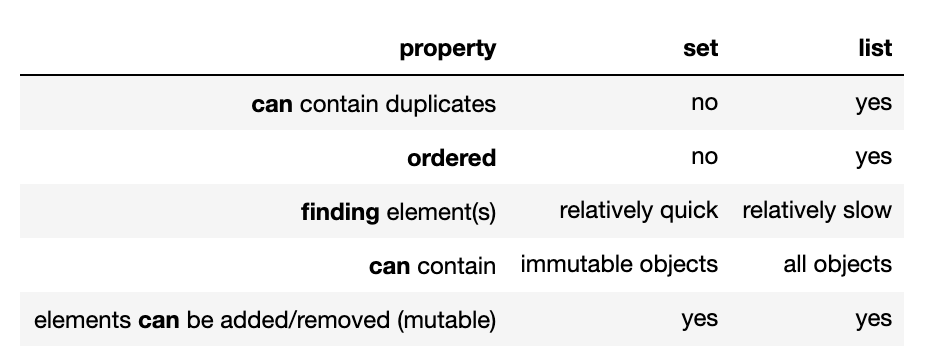


1. List:

* List.reverse( ) or List[ : : -1]
* ValueError: no index in the list (ordered)
* List can be empty and contain another list
* List: concatenation (+), repetition, membership, comparasion
* a.append(b) # Add item b to the end of a
* a.extend(c) # Add the elements of list c at the end of a
* a.insert(i,b) # Insert item b at position i
* a.pop(i) # Remove from a the i'th element and return it. If i is not specified, remove the last element
* a.index(x) # Return the index of the first element of a with value x. Error if it does not exist
* a.count(x) # Return how often value x is found in a
* a.remove(x) # Remove from a the first element with value x. Error if it does not exist
* a.sort() # Sort the elements of list a
* a.reverse() # Reverses list a (no return value!)

1. Set:

* Can be empty but NO duplicates
* Unordered
* Can only contain immutable items (string and integer, NO lists or sets > TypeError)
* Add, update, pop, remove
* SetA.union(SetB)
* SetA.intersection(SetB)

1. Comparison between SET and LIST: 

* Finding elements in set is quicker than list (no duplicates rule)

1. Tuple:

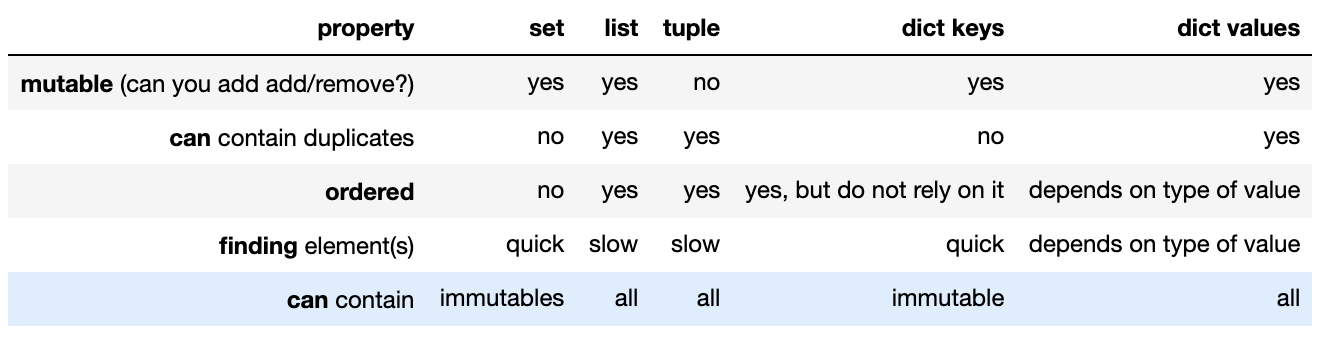
* Ordered items: index
* Immutable and mutable objects allowance (do not suggest any set or list): count
* Items cannot be added or removed
* Tuple can be empty
* Unpacking

1. Break, continue and pass in for looping:

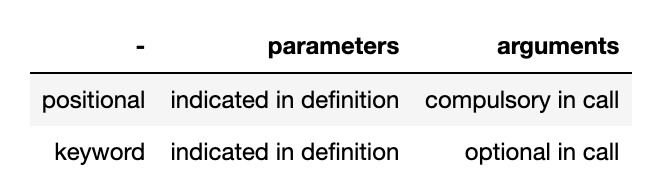
* Break: stop at the inquiry (incl inquiry)
* Continue: jump at the inquiry and continue looping

1. Dictionary:

* Keys in dicts are immutable and unique (NOT list…) > TypeError
* Dict can be empty
* Values can be anything
* Dict.items( ): tuple
* Dict.keys( ) or values( ):dict

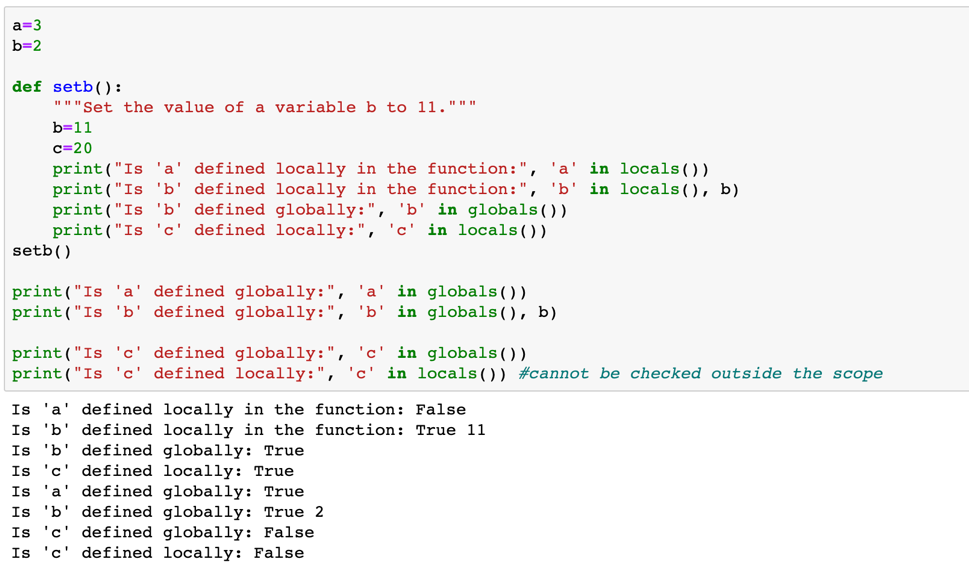


1. Define a function:



1. Assert function
2. Variable Scope:

* NameError: local scope is being called outside global scope



1. Open file > close:

* Open a file > read once for one time
* Read( ): read entire text in a file
* Readlines( ): read the content in a list, with (\n)
* Readline( ): return a line of content before (\n), line by line
* With open (x, “w”) as outfile:

Outfile.write (…)

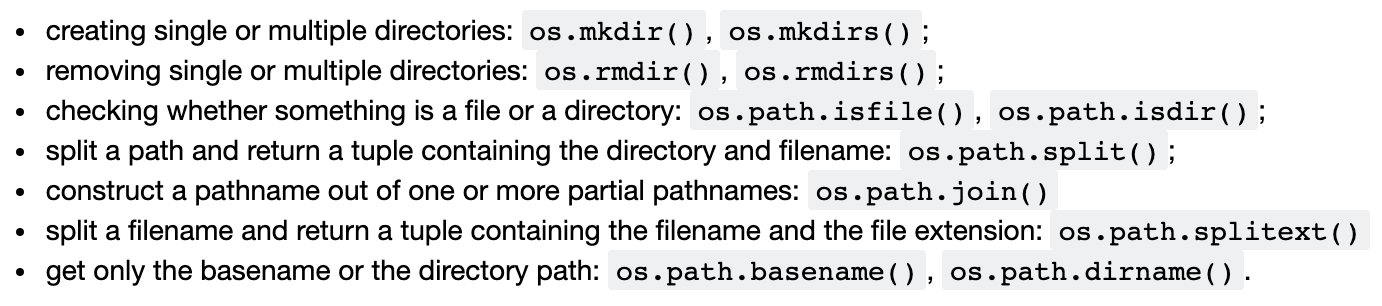
* Mode = “a” will append new content at the end of existing one, while mode = “w” will overwrite the content

1. Reading multiple files: os or glob

Glob:

* For filenames in glob.glob(“xxx/xxx/\*.txt”)
* For filenames in glob.glob(“xxx/xxx/a?.txt”)

Os:



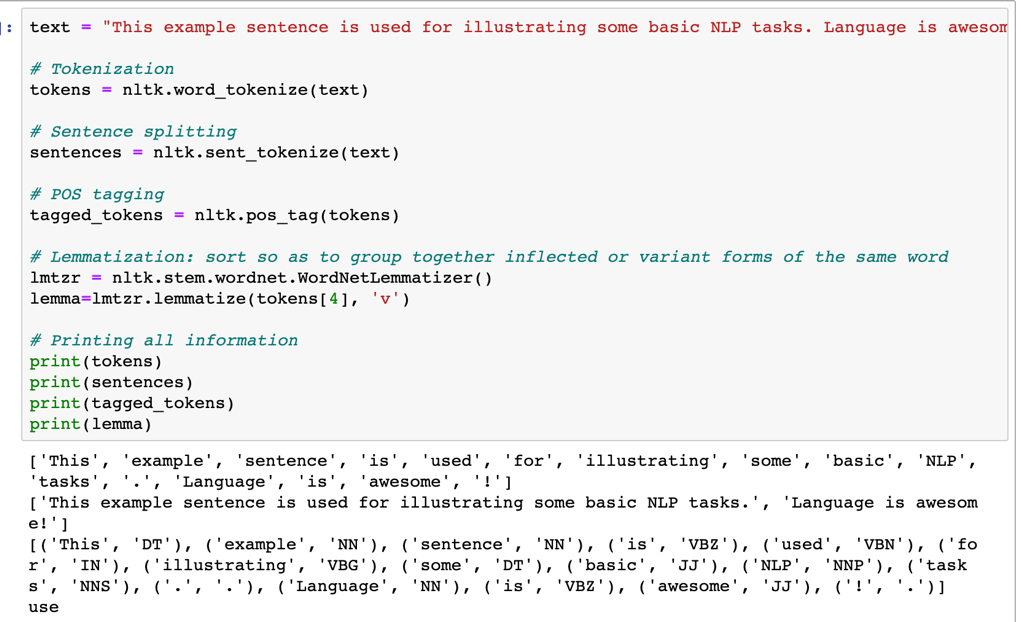
e.g. filepath = “../data/Charlie/Charlie.txt”

- dirname: “../data/Charlie”

- basename: “Charlie.txt”

- split: dirname, basename

- splitext: “../data/Charlie/Charlie”, “.txt”

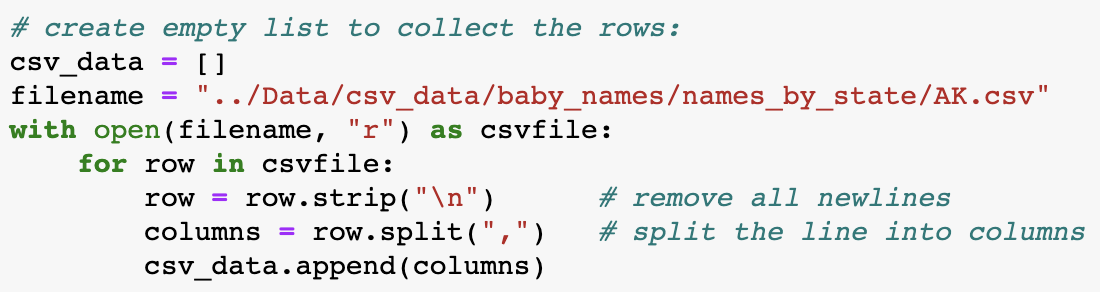
1. 

* Sent\_tokenize on a string > return a list
* Nltk\_pos\_tag > returns a list with tuples

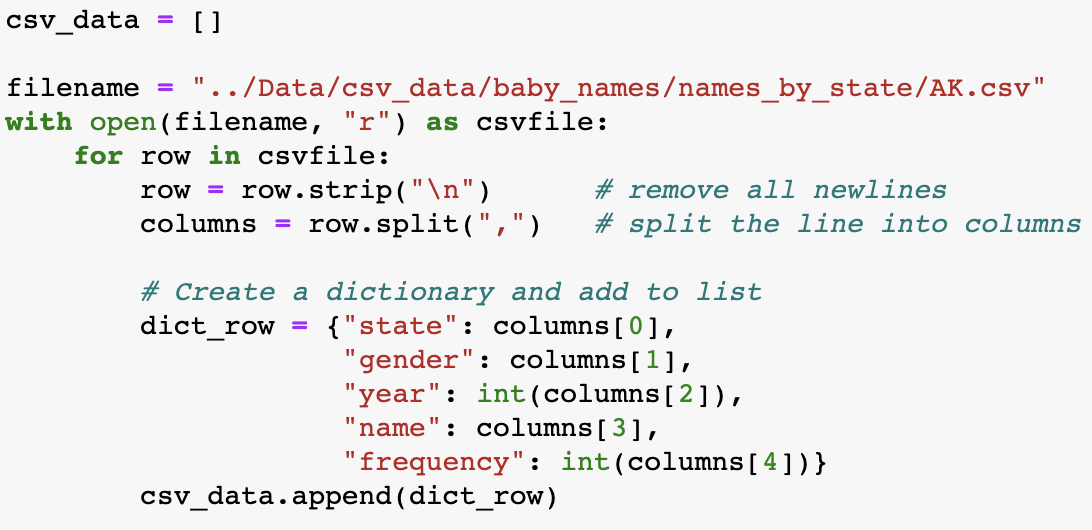
1. Data Format: CSV and TSV:

* Table formats
* CSV: comma separator
* TSV: tab separator (\t)
* CSV:

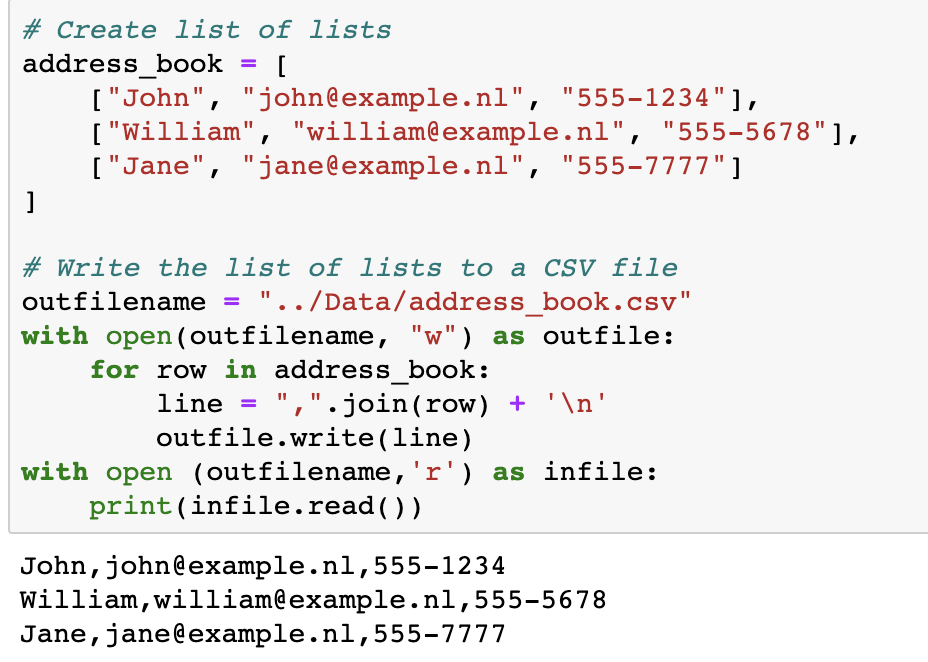
+ Reading rows as lists:



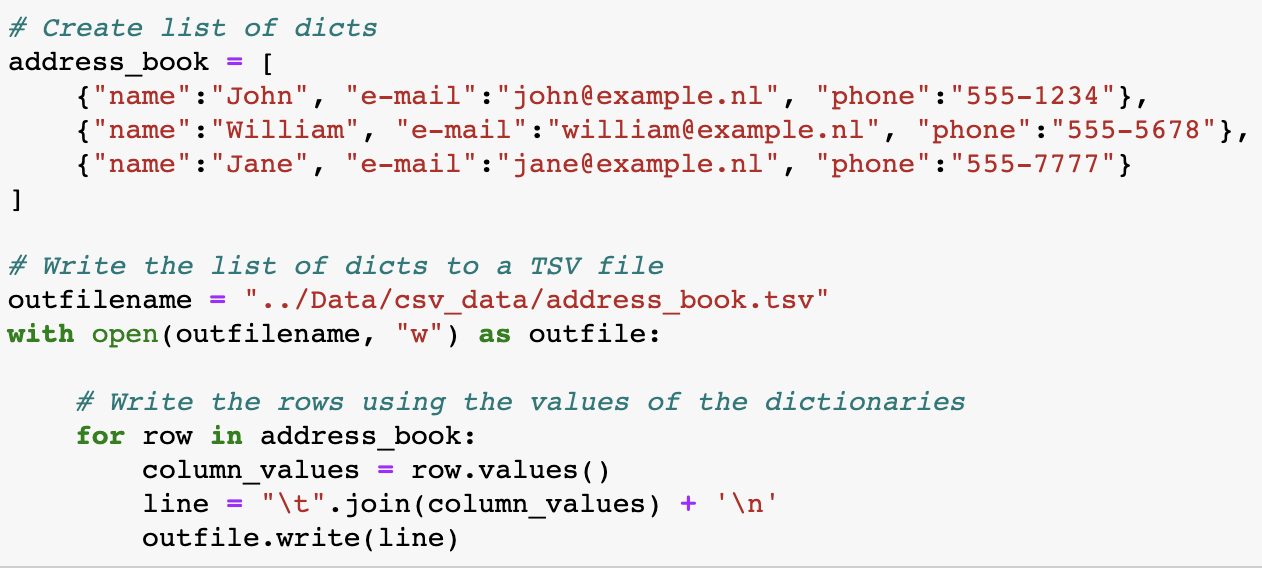
+ Reading rows as dicts:



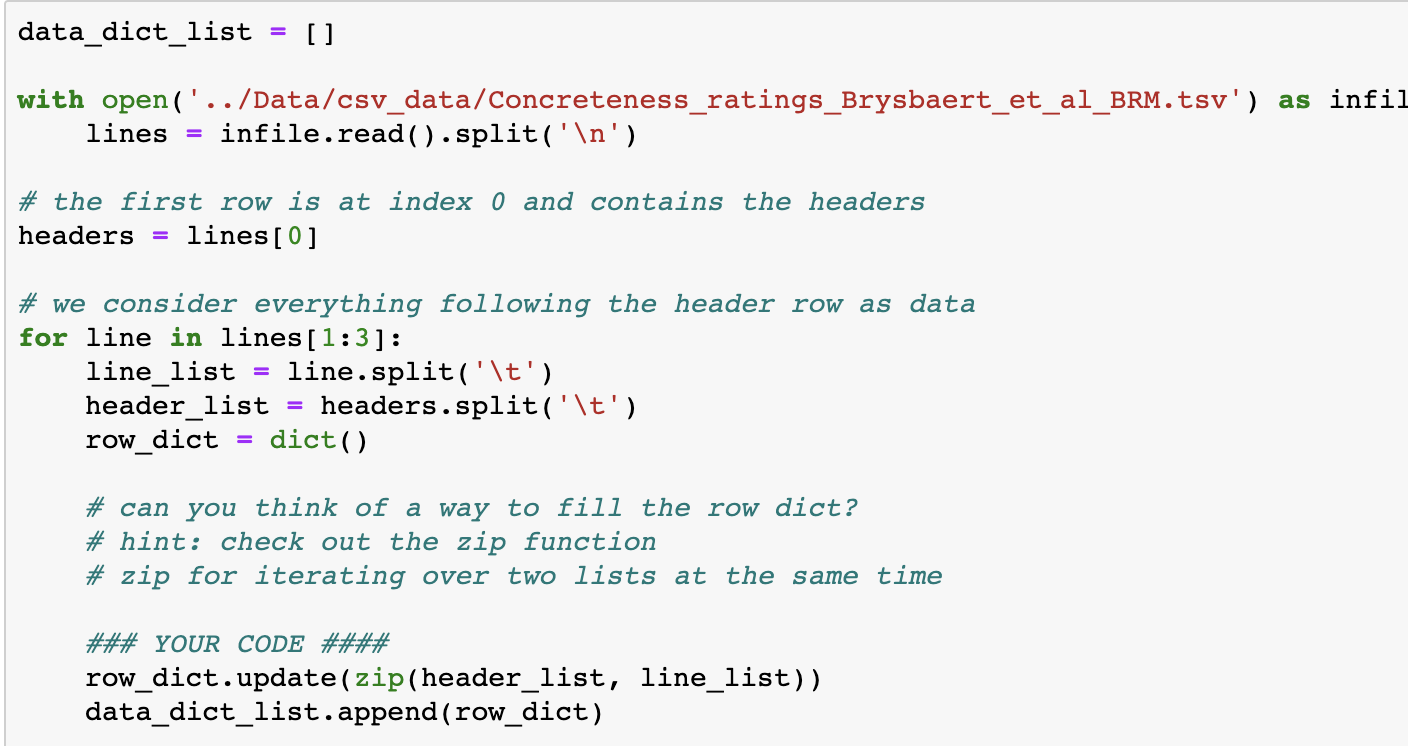
+ Writing rows as lists:



+ Writing rows as dicts:



* Writing headers:





1. Data Format: JSON

* Read JSON:

+ json.load( ) > return a dictionary

+ json.loads( ) > load a string, return a dictionary

* Write JSON:

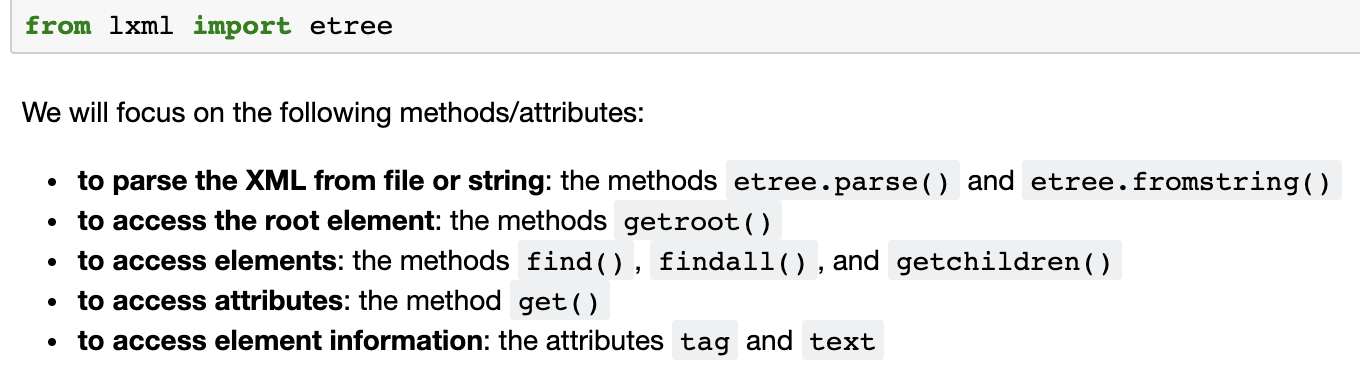
+ json.dump( ) > write a dictionary to a JSON encoded file

+ json.dumps( ) > convert a dictionary to a string

+ two useful keyword arguments: indent=4, sort\_keys=True

1. Data Forma: XML

* Some overview:



* Text, attribute and elements
* Some basic procedures:

- tree = etree.parse( pathway )

- root = tree.getroot( )

- print (etree.dump(root, pretty\_print=True)) > show elements

- Root.find ( ); Root.findall( ); Root.getchildren( )

- .text or .tag

- find(‘../../../../../’) > parse different layers, indicate the range