Converting Units Across Multiple Data Sources Using CSV Files

Mihran Simonian

12386294

Link to presentation, code, dataset and data generator:

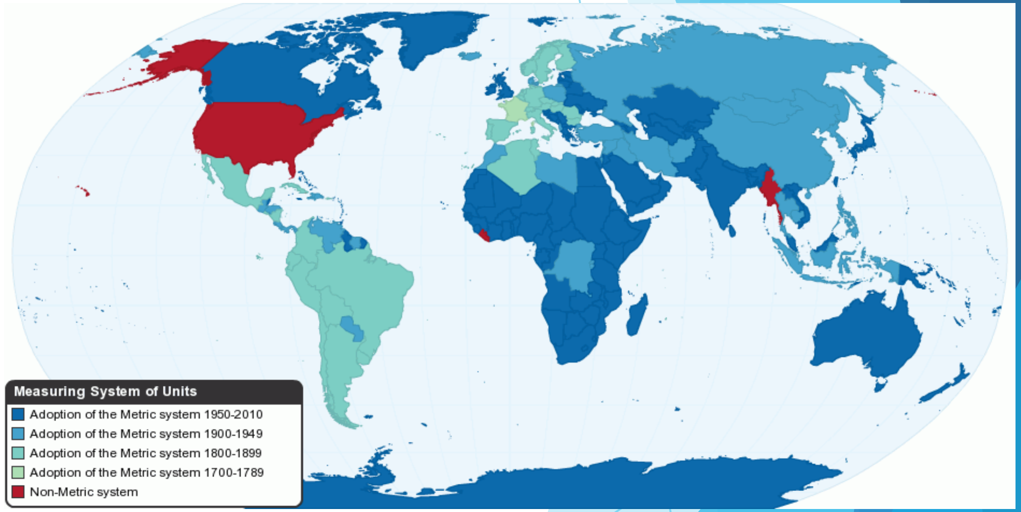
https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment

Introduction – What Are Measurement Units?

- Describe a measurement: Temperature / Distance / Weight etc.

- Different systems in the world, generally:
  - Metric system (SI)
    - Système international d'unites
    - 95% of the world
  - US customary system: USA
  - Burmese system: Burma

- Specific niches sometimes are applied, even today:
  - Imperial (UK)


Measuring System of Units
Adoption of the Metric system 1950-2010
Adoption of the Metric system 1900-1949
Adoption of the Metric system 1800-1899
Adoption of the Metric system 1700-1789
Non-Metric system

Source image: http://chartsbin.com/view/d12

Motivation

Link to presentation, code, dataset and data generator:

https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment

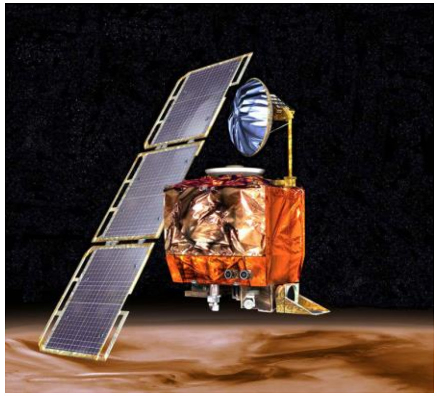
Motivation – Why Convert Units?

- Compare products
- Create a (technical) design
- Make decisions, e.g. procurement

- Some niches require;
  - Oil & Gas:
    - Large pipes measured solely in inches US customary units, except sometimes not when European vendor
    - Small pipes measured sometimes in US customary and sometimes in SI metric
  - Engine horsepower: bhp (imperial) and hp (SI)
    - “British manufacturers often intermix metric horsepower and mechanical horsepower depending on the origin of the engine in question.” (source: https://en.wikipedia.org/wiki/Horsepower)
  - Mechanical screwthread:
    - English (left-hand), Metric and US Customary (right-hand)

- In short; it’s quite a mess still

Motivation – When Things Go Wrong



- NASA Mars climate orbiter, 1999
- Mix-up between US Customary and SI
- Orbiter crashed due to misinterpretation of units (wrong calculations)
- Damage:
  - USD 125 million in 1999
  - No mars climate orbiting!

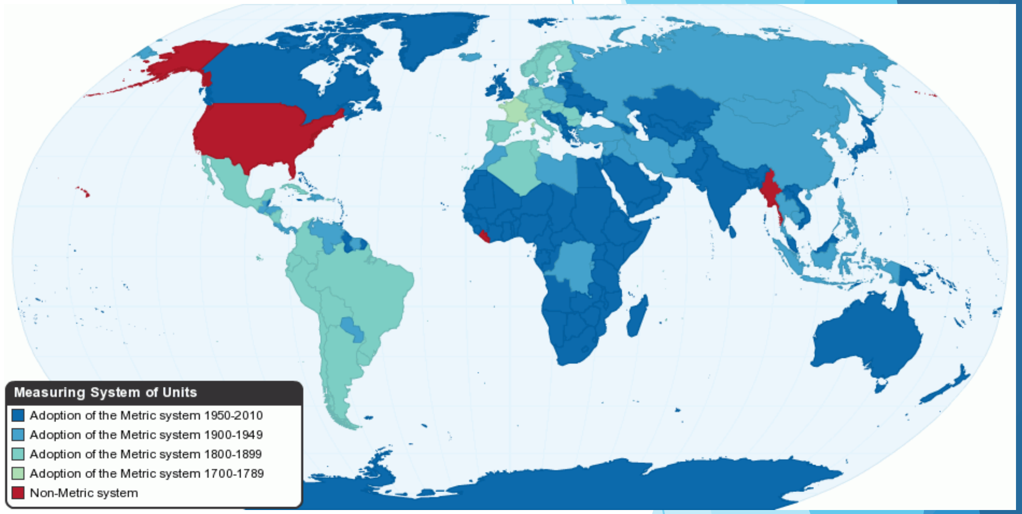
Remember the Mars Climate Orbiter incident from 1999?

Source costs: http://edition.cnn.com/TECH/space/9909/30/mars.metric/

Source image: https://solarsystem.nasa.gov/missions/mars-climate-orbiter/in-depth/
https://images.app.goo.gl/psJdWTtZb9C1uQkd7

Motivation - Situational Framework

- ERP system integration
  - Multiple vendors around the world
  - Use a single identifier for production identification
  - Have different units in place


Measuring System of Units
Adoption of the Metric system 1950-2010
Adoption of the Metric system 1900-1949
Adoption of the Metric system 1800-1899
Adoption of the Metric system 1700-1789
Non-Metric system

- Want a single unit representation

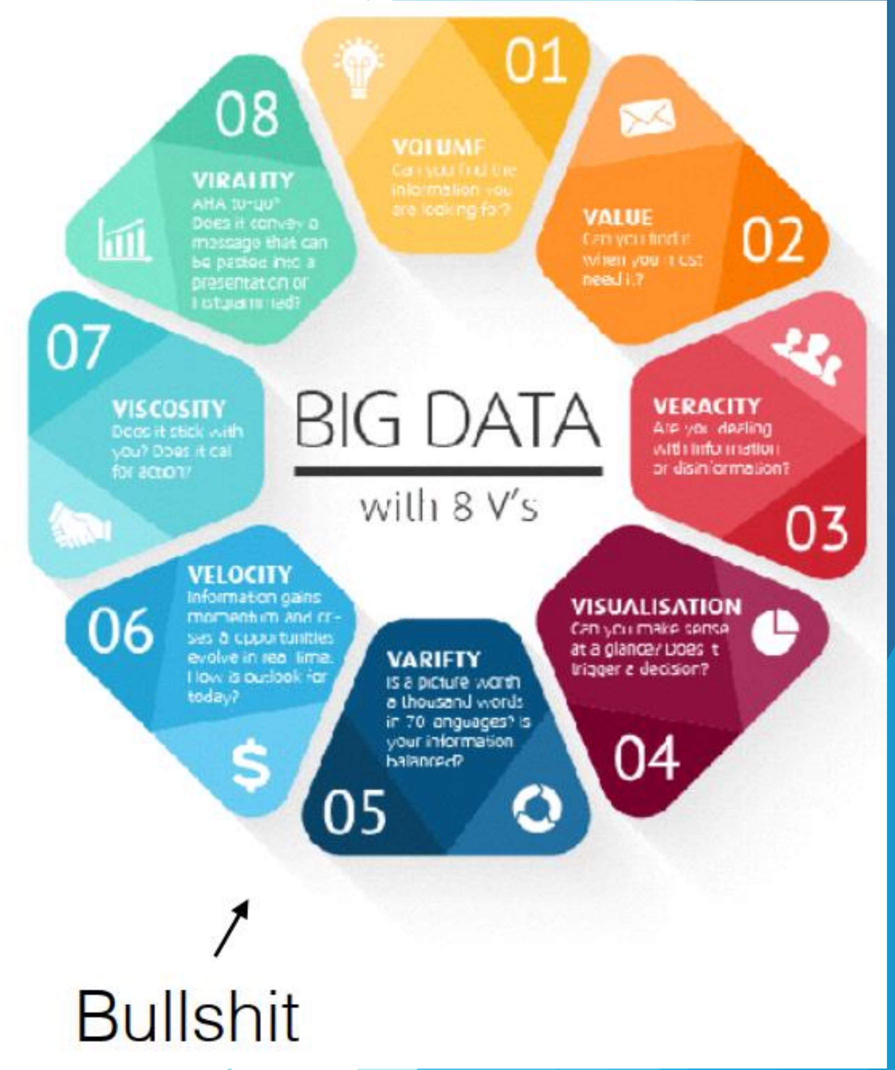
- Each vendor should be able to design their own translation list as well for local utilization (instead of SI use US customary)

- Easy adjustment and verification of unit conversion

Source image: http://chartsbin.com/view/d12

Motivation – Focus On Big Data

- The famous eight V’s
  - Some say there are too many V’s
- Focus lays on variety
  - Data types:
    - Strings
    - Floats
  - File conversion
  - Using files for dictionaries and databases
- Touch fields of volume and speed superficially


BIG DATA
with 8 V’s
01
VOLUME
Can you find the information you are looking for?
02
VALUE
Can you find it when you most need it?
03
VERACITY
Are you dealing with information or disinformation?
04
VISUALISATION
Can you make sense at a glance? Does it trigger a decision?
05
VARIETY
Is a picture worth a thousand words in 70 languages? Is your information balanced?
06
VELOCITY
Information gains momentum and crises & opportunities evolve in real time. How is outlook for today?
07
VISCOSITY
Does it stick with you? Does it call for action?
08
VIRALITY
AHA to-go? Does it convey a message that can be pasted into a presentation or Instagrammed?
Bullshit

Source image: Lecture 1, Block 3, University of Amsterdam, Big Data Class

Existing Methods

Link to presentation, code, dataset and data generator:

https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment

Existing Methods – Python Libraries

- Multiple libraries exist
- Either they are not sufficient, or they are too sufficient
  - Some abbreviations in measurement systems are used for multiple ‘subjects’, e.g. : R
    - Rockwell surface hardness / Radius / Rankine temperature
  - Some packages require the special characters: ° in temperatures, not always accepted in your csv container
- Error codes frequently arose
- Main issue;
  - it requires the coder to know what he is doing
  - it requires coding experience to verify/adjust the operation

Proposed Method

Link to presentation, code, dataset and data generator:

https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment

Proposed Method – Requirement: Use Multiple Files

- Multiple files arrange the unit conversion
  - Unit conversion file - CSV
  - Input: database file – CSV
  - Output: cleaned database
  - Output storage:
    - Csv
    - Sql
    - Hdf
    - Parquet
    - Feather

- This allows non-programming users to verify and adjust the unit conversion

- CSV is easy to understand, works with nearly any system in the world (import/export) and is supported even by Windows Notepad

Implementation

Link to presentation, code, dataset and data generator:

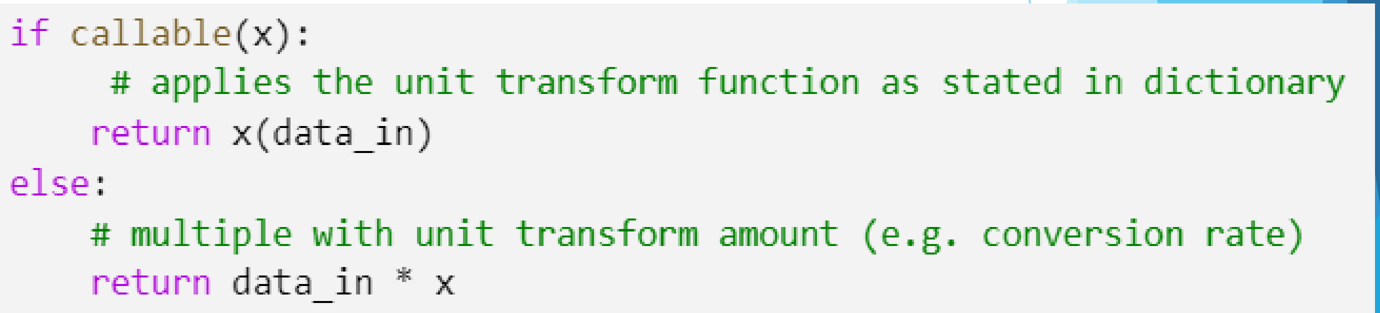
https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment

Implementation – General Steps

- 1. Import
  - Database
  - Unit conversion list
- 2. Process
  - Convert to Pandas Dataframe
  - Clean each data entry (decimal points, strings, remove columns etc.)
  - Perform unit conversion
- 3. Output
  - Store Pandas dataframe in data container

Implementation – Back to Basics: How to Convert A Unit?

- 1. Apply formula:
  - E.g. Fahrenheit to Celsius:
    - $T_{(^\circ\mathrm{C})} = (T_{(^\circ\mathrm{F})} - 32) \times 5/9$

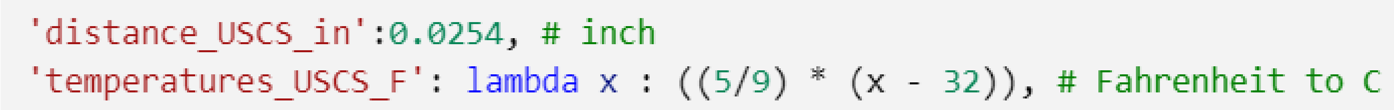


```
if callable(x):
     # applies the unit transform function as stated in dictionary
    return x(data_in)
else:
    # multiple with unit transform amount (e.g. conversion rate)
    return data_in * x
```

- 2. Multiply by static number
  - Inches to Centimeter:
    - $D_{(cm)} = d_{(\mathrm{inches})} * 2.54$

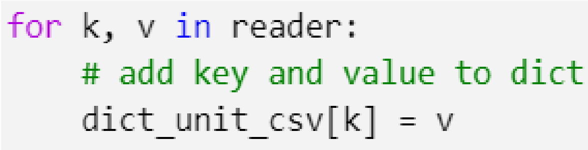
Proposed Method – CSV Unit Conversion?

- Dictionary can contain both methods for unit conversion (static and formula)



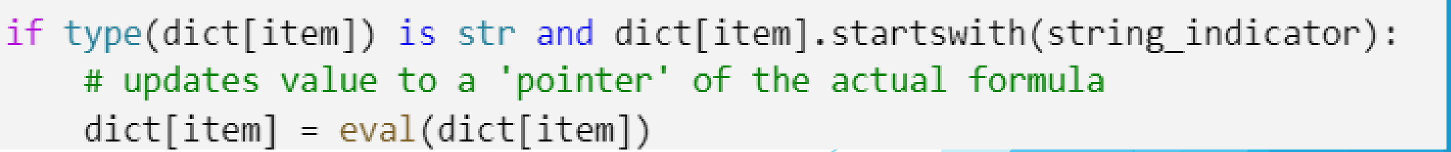
```
'distance_USCS_in':0.0254, # inch
'temperatures_USCS_F': lambda x : ((5/9) * (x - 32)), # Fahrenheit to C
```

- A conversion number



```
for k, v in reader:
    # add key and value to dict
    dict_unit_csv[k] = v
```

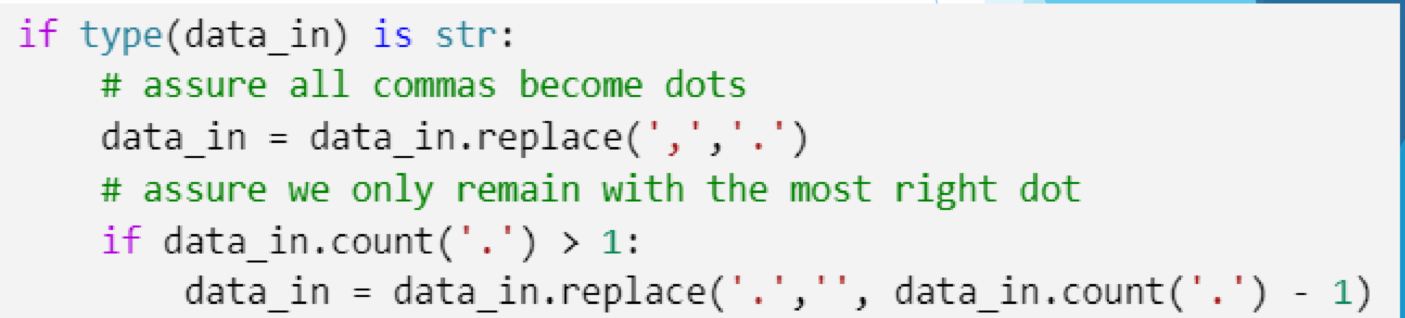
- Special case: a formula
  - Read a string to memory location
  - Update dictionary and create a pointer to the memory location



```
if type(dict[item]) is str and dict[item].startswith(string_indicator):
    # updates value to a 'pointer' of the actual formula
    dict[item] = eval(dict[item])
```

Proposed Method – Dots, Decimal Points and Thousand Separators

- Countries use different systems
  - Choose to use decimal point
- Software can create thousand separators
  - Excel
    - US: commas
    - Netherlands: dots
- Sometimes your values are stored in strings
  - Try: float(number_which_is_now_in_string)
- Error handling included



```
if type(data_in) is str:
    # assure all commas become dots
    data_in = data_in.replace(',','.')
    # assure we only remain with the most right dot
    if data_in.count('.') > 1:
        data_in = data_in.replace('.','', data_in.count('.') - 1)
```

Proposed Method – Error Handling

- Error handling included in functions

- Main case: the variable is not the correct datatype
  - String is delivered expect a float
  - Boolean delivered expect a float

- Solution:
  - Return string
  - Raise RuntimeError with datatype given and custom message (included in each function

- Using Booleans to determine whether to use error handling or not

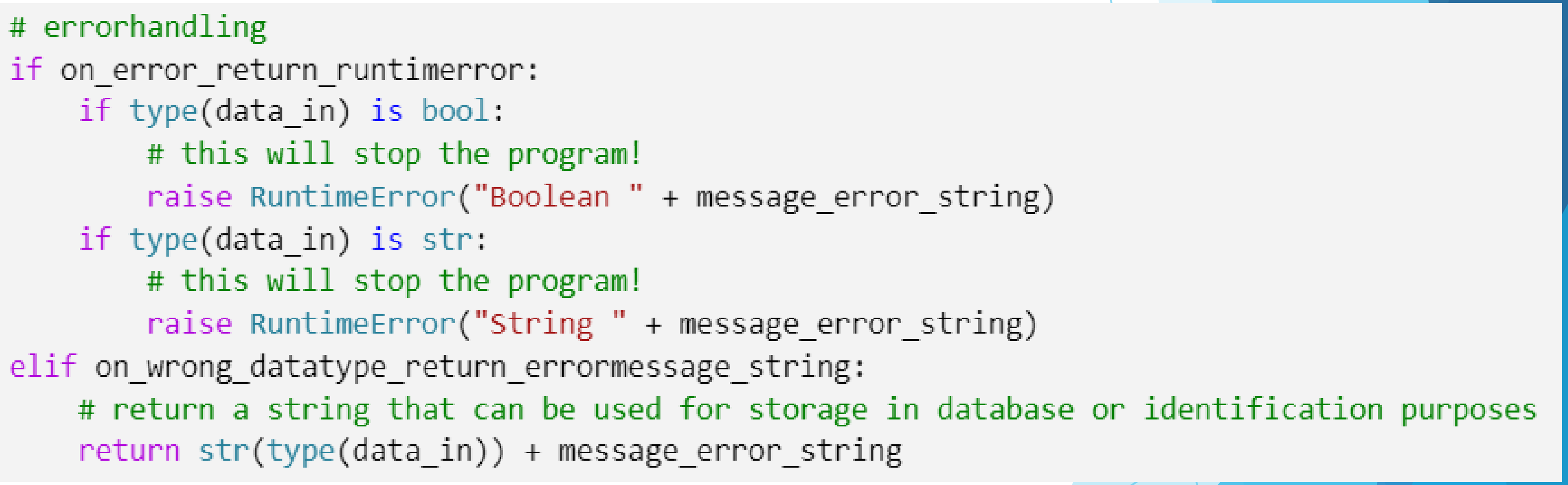
Proposed Method – Error Handling

- Def function(input, return_error_message=False, return_runtimeerror=False):

- If input is not what I expect:
  - Try:
    - Fix It if possible
  - Except:
    - if return_runtimeerror:
      - **Raise** runtimerror(error message)
    - If return_error_message:
      - **Return** errormessage
  - Return function performed on input # everything is okay and we can return the input

Proposed Method – Error Handling

- Def function(input, return_error_message=False, return_runtimeerror=False):
- If input is not what I expect:
  - Try:
    - Fix It if possible
  - Except:
    - if return_runtimeerror:
      - **Raise** runtimerror(error message)
    - If return_error_message:
      - **Return** errormessage
  - Return function performed on input # everything is okay and we can return the input



```
# errorhandling
if on_error_return_runtimerror:
    if type(data_in) is bool:
        # this will stop the program!
        raise RuntimeError("Boolean " + message_error_string)
    if type(data_in) is str:
        # this will stop the program!
        raise RuntimeError("String " + message_error_string)
elif on_wrong_datatype_return_errormessage_string:
    # return a string that can be used for storage in database or identification purposes
    return str(type(data_in)) + message_error_string
```

Proposed Method – Error Handling

- Return string:
  - Can be stored in the database for error tracking

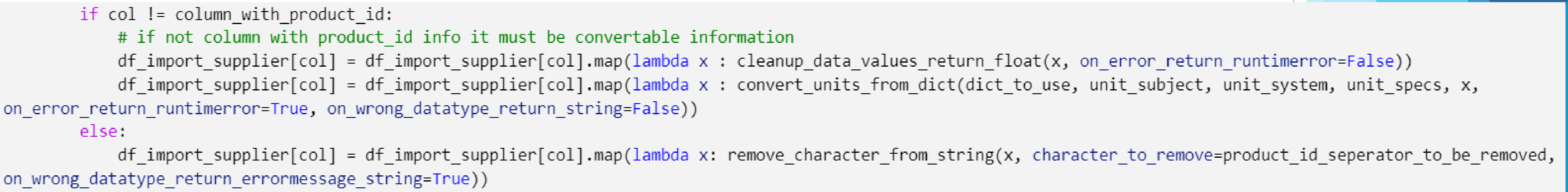
- Return runtimeerror:
  - Will stop running the program, so much more useable for programmers

- Runtime error has been included as it provides better tracking of error (as you can see the codelines)

Proposed Method – Perform Conversions

- Import -> Dataframe

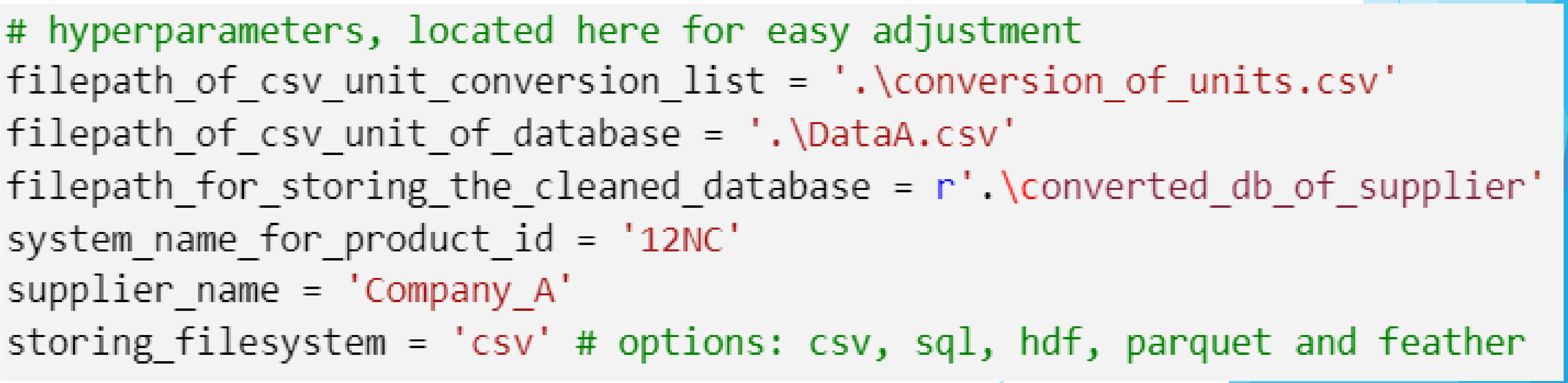
- Perform map & lambda operation on the columns in the dataframe



```
        if col != column_with_product_id:
            # if not column with product_id info it must be convertable information
            df_import_supplier[col] = df_import_supplier[col].map(lambda x : cleanup_data_values_return_float(x, on_error_return_runtimerror=False))
            df_import_supplier[col] = df_import_supplier[col].map(lambda x : convert_units_from_dict(dict_to_use, unit_subject, unit_system, unit_specs, x,
on_error_return_runtimerror=True, on_wrong_datatype_return_string=False))
        else:
            df_import_supplier[col] = df_import_supplier[col].map(lambda x: remove_character_from_string(x, character_to_remove=product_id_seperator_to_be_removed,
on_wrong_datatype_return_errormessage_string=True))
```

Proposed Method – Store To Container

- Use built-in pandas dataframe solutions
- Generally only a few hyperparameters
- Easily iterateable, so we can upscale to automate our conversion method



```
# hyperparameters, located here for easy adjustment
filepath_of_csv_unit_conversion_list = '.\conversion_of_units.csv'
filepath_of_csv_unit_of_database = '.\DataA.csv'
filepath_for_storing_the_cleaned_database = r'.\converted_db_of_supplier'
system_name_for_product_id = '12NC'
supplier_name = 'Company_A'
storing_filesystem = 'csv' # options: csv, sql, hdf, parquet and feather
```

Upscaling To Big Data

Link to presentation, code, dataset and data generator:

https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment

Upscaling To Big Data - Storage

- Pandas Dataframe offers storage to popular formats:
  - CSV
  - SQL
  - Hdf
  - Parquet
  - Feather

- CSV: ~11 MB
- Parquet (compression: 5): ~3 MB
- Further compression (zip, gzip etc.) possible

Upscaling To Big Data - Processing

- Using variable translation lists and databases, only functions need to be copied to individual nodes
- Relatively simple and self contained functions
- Parallelization is therefore easily possible, even without the need of using advanced techniques
- The whole import and transformation process can run on a single node
- A network of nodes can process multiple new supplier databases
- Merging to an existing database depends on the applied techniques

Conclusion

Link to presentation, code, dataset and data generator:

https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment

Conclusion – A Very Unique System

- Combine loose files with specific software
- By making software as generic as possible, this can be used for many instances
- A relative quick system
- Universally deployable
- Requires little to no additional libraries (depends on storage output, code provides for installation of libraries)

Discussion

Link to presentation, code, dataset and data generator:

https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment

Discussion – Humans Remain Involved

- Humans are able to adjust csv files, potentially ruining the system
- We will always need a human verification
- System was specifically designed to tailor the human side of the conversion; being able to use human’s capabilities in interpreting information and thus assign a new measurement unit system
- Security implications as people might create functions that you don’t want to execute (formatting a drive, deleting files etc.)

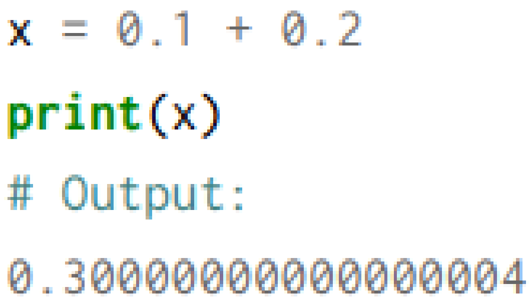
Discussion – Computer Imprecision

- Computer bits allow only 1 or 0 to be stored
- Numbers in essence are infinite but bits create a limitation
- Due to bit size, we will always remain with an imprecision

- For simple calculations not an issue
- For the more advanced conversions much more important

- For most cases not important
- For NASA very important

- With each conversion
  - Loose a bit of information
  - Create misinformation



```
x = 0.1 + 0.2
print(x)
# Output:
0.30000000000000004
```

End

Link to presentation, code, dataset and data generator:

https://github.com/mihransimonian/Uva-Block3-Big_Data-Assignment