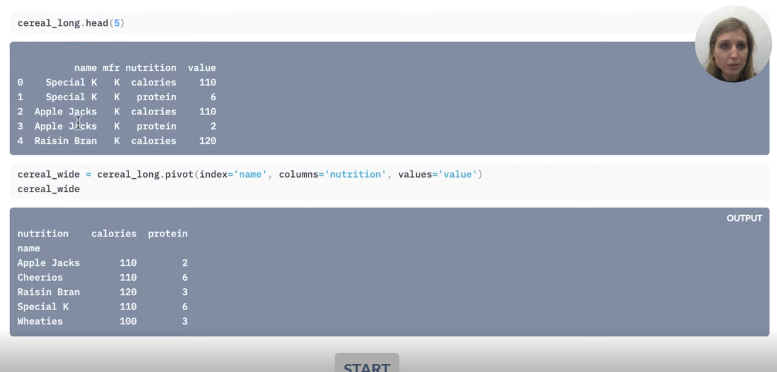
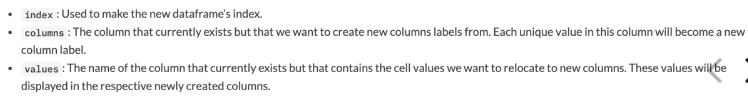
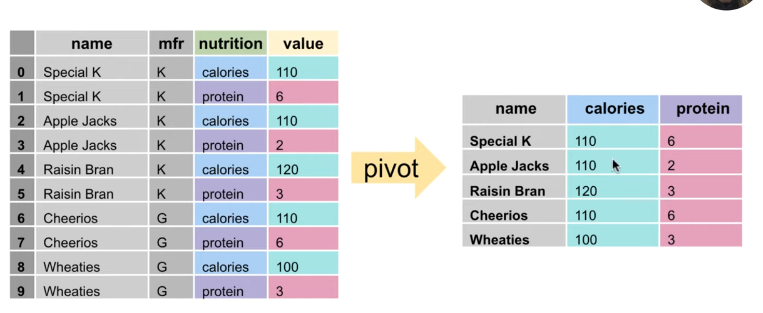
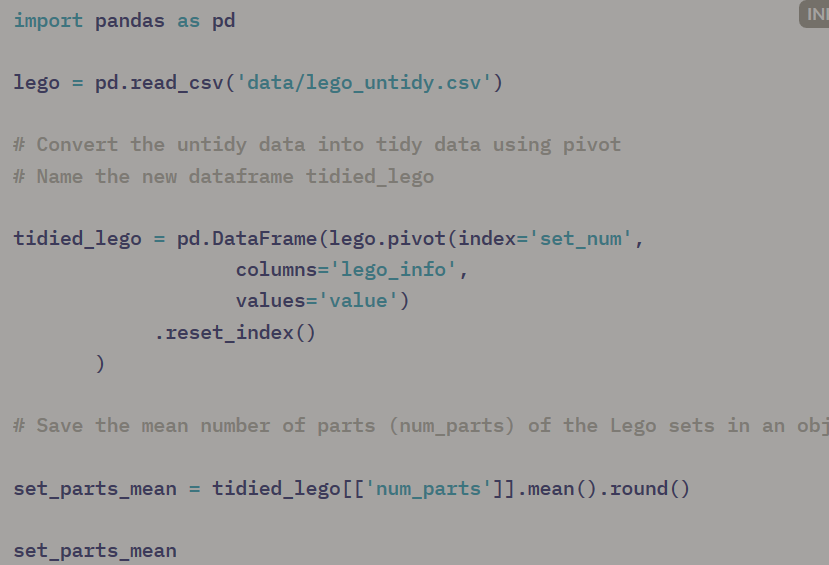
Python Module 3 Notes

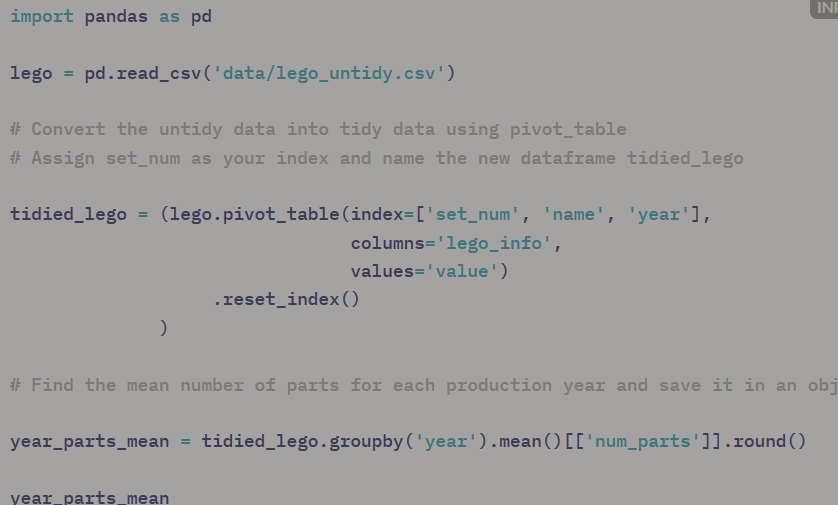
Tidy data

* Creates standard across multiple analysis tools
  + 3 criteria:
  + 1-- each row is a single observation
  + 2—each variable is a single column
  + 3—each value is a single cell
* Statistical questions and tidy data
* Reshaping with Pivot
  + Used for transforming long dataframes into wider ones
  + 
  + 
  + 
  + Index is reassigned to the name column
  + However, we lost the mfr column
  + Use .reset\_index(), then do the following:
  + 
    - .rename\_axis(‘new name’, axis=’columns or rows’ (depending on what you’re renaming)
  + Example

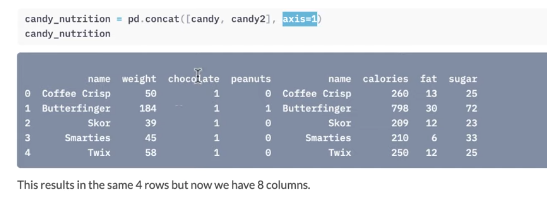


* Reshaping with Pivot Table
  + Use .drop to remove rows along with axis and index numbers to specify which to remove

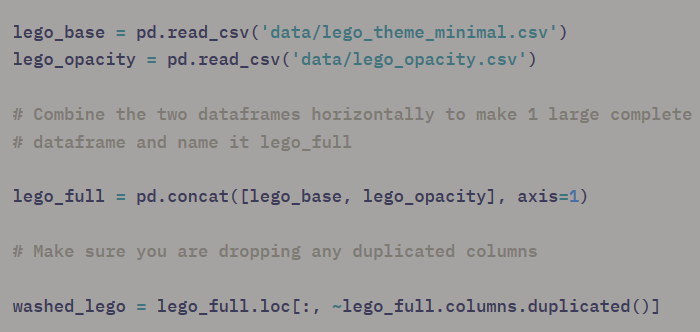
Example with multiple index names



* Reshaping with Melt
  + Opposite of pivot (wide 🡺long dataframe)
  + 
  + Id\_vars = identifying column and ones you wish to keep
  + Value\_vars= columns containing values (amalgated into one column)
  + Var\_name= name of amalgamated value columns
  + Value\_name= column containing values from the original value columns
* Concatenation
  + Pd.concat()—used for combining two dataframes
  + .merge()



* Example above: candy and candy2= dataframes you wish to combine; axis=1 = columns of dataframe (axis=0 is for rows) – glues them along column axis
  + To eliminate duplicate columns: use .loc,~, and .duplicated()
  + 
* To vertically combine dataframes: axis=0
* Remove index using drop()
  + 
* Example



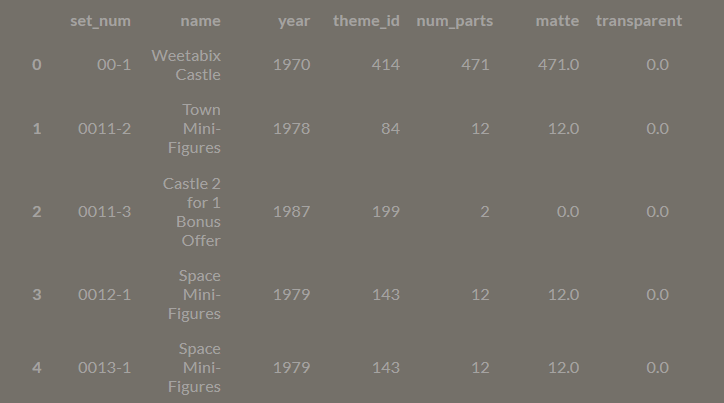
* To make a new column from two existing ones and to sort the dataframe:

:

lego\_details = (washed\_lego.assign(total\_pieces = washed\_lego['matte'] + washed\_lego['transparent'])

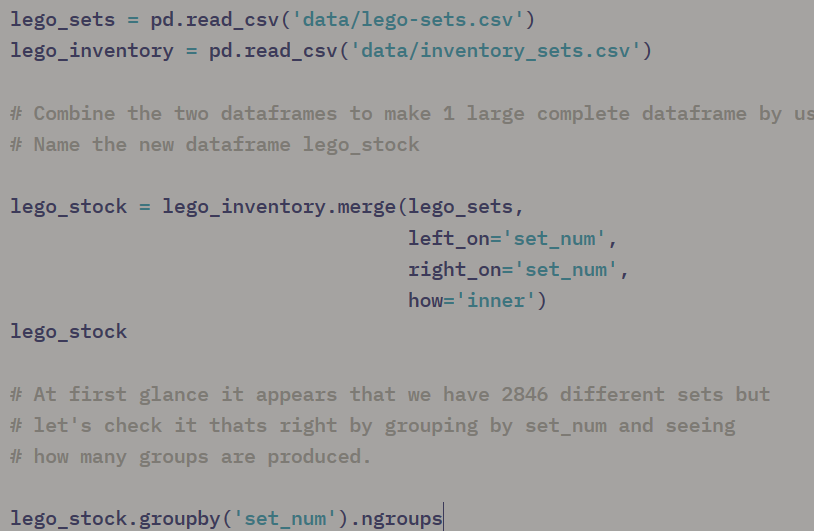
.sort\_values('total\_pieces', ascending=False)

)

* 
* What does shape function do? ---

Merge—gives more precision and options when joining dataframes

* Can only combine dataframes horizontally
  + Identify columns to merge
  + 
  + Decide left dataframe (e.g. ‘candy’) – candy.merge
    - Decide right dataframe (e.g. candy2m)—candy.merge(candy2m, .. )
    - .merge() needs a column which exists in both dataframes which contain common values
      * Left\_on—left dataframe
      * Right\_on—right dataframe
    - 
    - However, you have to watch out since the merge function will drop columns not present in both tables
      * To avoid this, use how=
        + E.g. 
      * How commands: inner, outer—returns everything(including blanks), left, right
        + Outer – returns everything, leaving blank cells with NaN
        + Left—outputs rows in left dataframe;omits anything exclusive to right dataframe
        + Right – outputs rows in right dataframe;omits anything exclusive to left dataframe
      * Indicator—used to indicate what dataframe data originates



**MODULE 4**

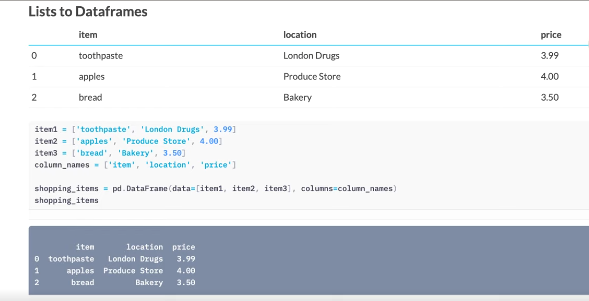
Values and Objects

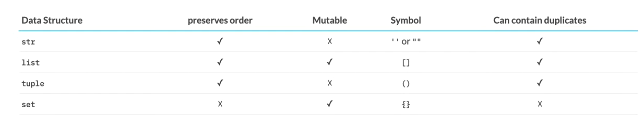
* Numbers (integers or floats) often don’t require any brackets around them
* NaN values—not a number—special value used to represent missing values in data
  + Still considered a number (float)
  + Cannot be converted to other values
* Nonetype – it’s own value type (None)
  + Useful for saving a variable w/o inputting any info
* Booleans—True and False values
* Strings—text
  + Sequence of characters
    - Use ‘’ and “”
    - If strings contain “” and apostrophes, use “”” “””
  + String verbs
    - Len()
    - .upper() – changes cases to uppercase
    - .lower() – lowercase
    - .count() – counts occurrences (case sensitive)
* Casting – turn something into a certain data type
  + E.g. int to float:
    - Number\_of\_baloons = float(5)
  + E.g. 2. Int to a str:
    - Number\_of\_balloon\_strings = str(5)
  + E.g.3. float to an int:
    - Number\_of\_balloons= int(4.99)
      * Number\_of\_balloons
        + 4

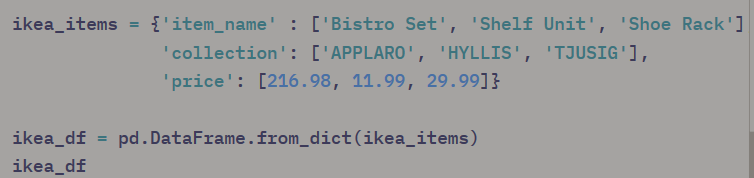
Exercise 1: count all occurrences of “let it be” in the titular Beatles song:

Letitbe\_count= lyrics.lower().count(‘let it be’)

Structure lists, tuples and sets

* .split() – splits sentence into multiple words
  + E.g. 2. Sentence.split(e) : splits sentence at every ‘e’ in sentence
  + [] – refers to list in this context
    - List is like a string, but more flexible
      * Can contain multiple data types within
  + To slice list by lists by elements
    - My)list[1] – gets first element
* Mutable vs immutable
  + Mutable—can be modified (e.g. lists)
  + 
  + [1] –selects 2nd list
  + [2] – selects 3rd element of 2nd list
  + =sandal—changes ‘Shoe’ to ‘Sandal’
  + Immutable—cannot be changed
    - E.g. strings
      * Changing an element within a string will result in a typeError
    - List verbs
      * .append() – adds something to end of list
* Lists to Dataframes
  + Can make a df from scratch using lists
  + E.g. 
  + Uses lists and column\_names to make a df
* Tuples
  + Like lists but: 1. use () to make them and 2. immutable
* Sets
  + Like lists but: 1. Unordered (no element 0, 1, etc.), and 2. Values contained are unique and 3. Use {}
  + Can still add to sets using .add()
* Summary



* Dictionaries
  + E.g. 
  + House[‘price’] – looks up house price
* Add a new key
  + E.g. house[‘bed monster’] =True
* Access all keys with .items()
* Make dataframes from dictionaries
  + Pd.DataFrame.from\_dict()
  + Can also do it by row by row:
    - 
  + Dictionary to Dataframe Example:
    - 
* DataFrames, series and column dtypes
  + Df = a collection of columns
  + Panda series—a list with a name
    - Building block of a df
  + Dtype—can be numeric (float64, int64) or non-numeric (object, bool, datetime64, timedelta[ns])
  + Non-numeric dfs
    - Object—default dtype
      * Contains strings
    - Bool
      * Bool
* Python Operations
  + Strings and math
    - Can sum, but not multiply or subtract
    - Multiply by an integer will repeat the string however many times
  + Bool—true=1; false=0
  + Lists—sum concatenates; can’t subtract or multiply
  + Strings and lists cannot be concatenated
* Operations with Columns
  + .assign() –assign something to the dataframe
  + E.g.
* Object columns—sum concatenates; .mean() doesn’t work
* Bool – cereal[‘hot’].mean() – works! Since True=1
  + % of true values
* If you want sum of values in a row
  + Cereal.loc[:, ‘protein’: ‘carbo’].sum[axis=1]
  + To add new column with this data:
  + 
* Splitting a column
  + String split
  + .str.split() – splits a whole column
  + 
    - Expand= true – splits values into separate columns
  + Rename columns: 

Detailed Example: canucks = pd.read\_csv('data/canucks.csv')

# Split up the column "Birth Date" into 3 separate columns named Birth\_Day, Birth\_Month and Birth\_Year.

# Name this new dataframe birthdate\_df

**birthdate\_df = (canucks['Birth Date'].str.split('-', expand=True)**

**.rename(columns={0:'Birth\_Day',**

**1:'Birth\_Month',**

**2:'Birth\_Year'}) )**

# Save these as columns in the canucks dataframe as dtype int

**canucks = canucks.assign(Birth\_Day=birthdate\_df['Birth\_Day'].astype('int'),**

**Birth\_Month=birthdate\_df['Birth\_Month'],**

**Birth\_Year=birthdate\_df['Birth\_Year'].astype('int')**

)

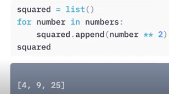
**Module 5**

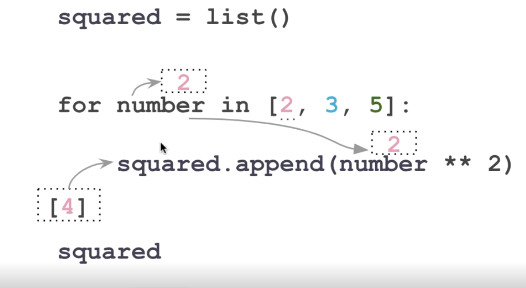
Making Choices with Conditional Statements

* How to ask Python to make decisions using conditions
* E.g. my\_name= “Hayley”
  + If my\_name == ‘Hayley”:
  + Print(“My name is Hayley too!”)
* Elif my\_name.lower() == ‘totoro’:
  + Print (‘Interesting, I love that movie’)
* Else:
  + Print(‘Sod off’)
* In line conditional statement example:
  + cups\_of\_tea =[2,4,3,5,2,3]
  + tea\_amount= sum(cups\_of\_tea) if type(cups\_of\_tea) is list else ‘cannot sum’

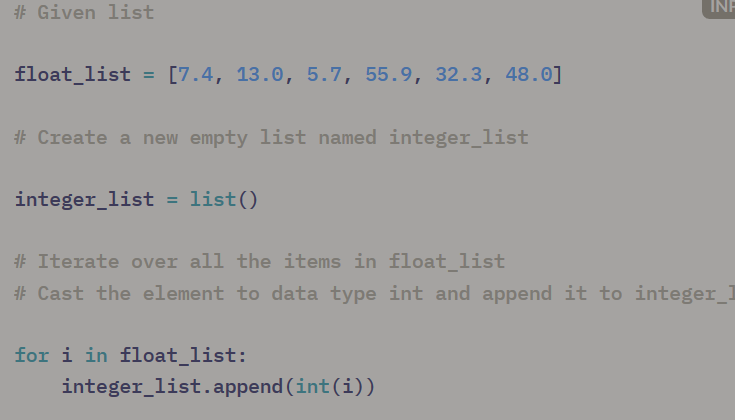
Repeating iterations

* loops

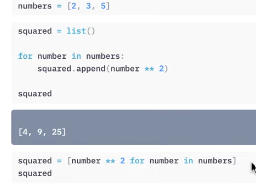
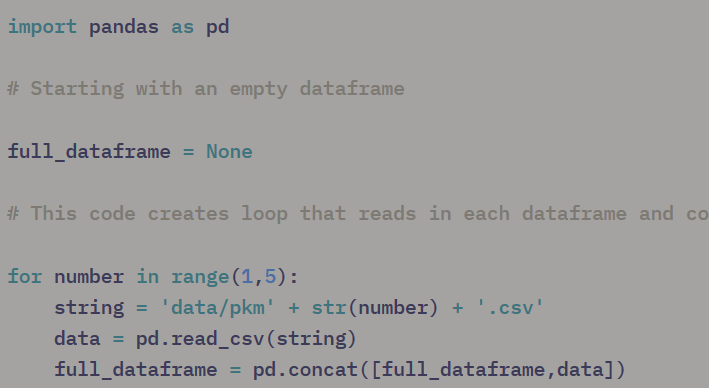
e.g. 

e.g. 2 

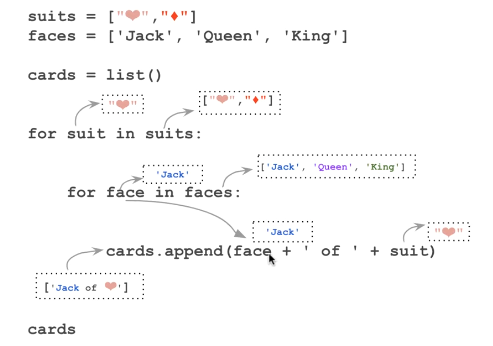
e.g. 3



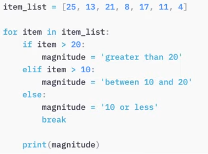
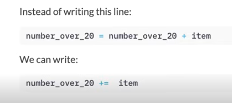
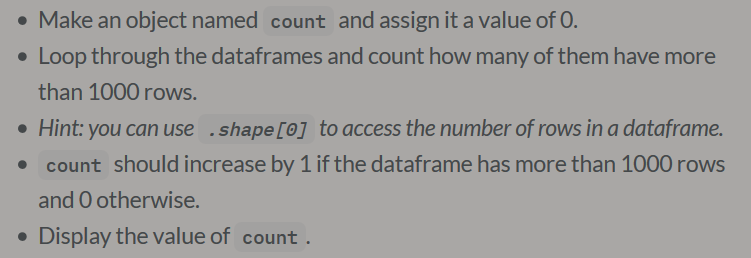
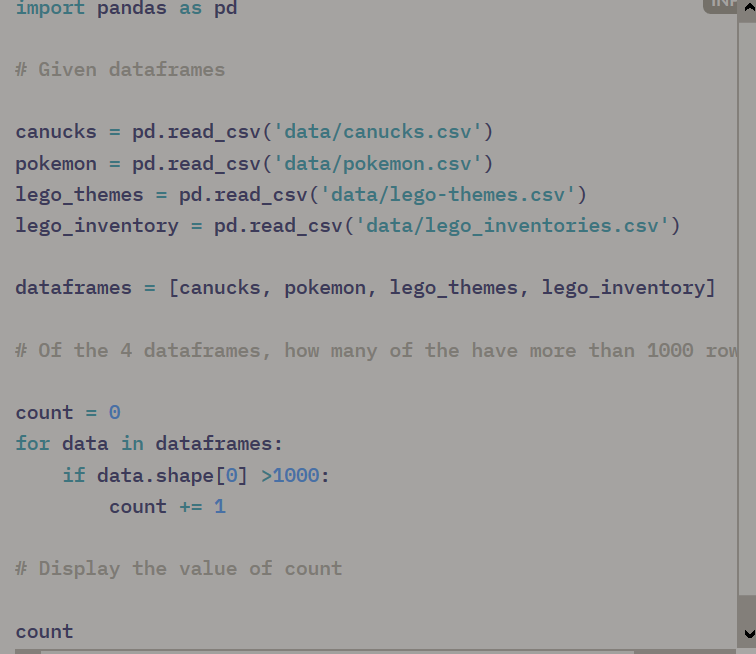
Range, dictionaries, and comprehension

* Range—generates collection of integers in sequence
  + Can specify start and end, and skip value)
  + Only works for integers (floats don’t work)
* Looping in a dictionary
  + .items()
  + Specify one for the key and one for the dictionary value
  + 
* For numerical
  + E.g. for I in range(50, 10, -4)
  + Print(i \*\*2)
    - 50=start value, 10=end value, -4= skip value (goes down by 4)
* Comprehensions
  + Allow you to write a list using one line of code
  + E.g. 
  + Dictionary comprehensions
    - 
  + A loop with a dataframe:
  + 

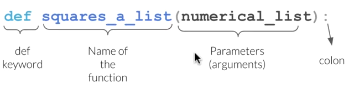
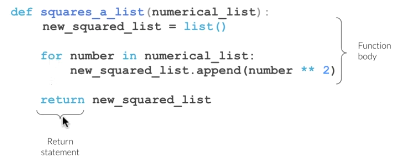
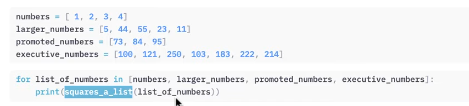
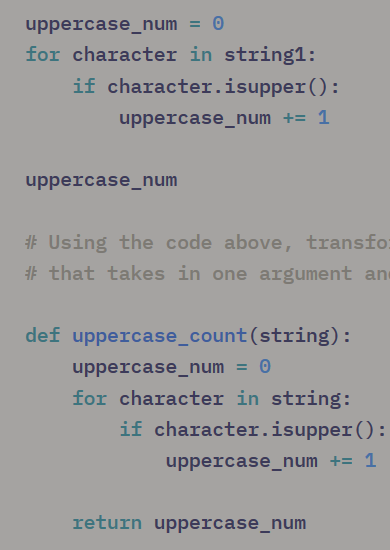
Nested Loops

* Second loop within the first loop
  + 

Repeated iterations with conditions

* E.g. Conditional statements within a for loop
* 
* Break—stops loop from continuing
* A stylising side point
  + 
* E.g. using dataframes:
* 
* 

Functions

* How to do loops w/ multiple lists?
* Defining a function
  + Def—anything indented after this belongs to a function
  + 
  + 
  + 
* When do we design a good function?
  + A lot comes down to personal opinion
* Example: 
* Example 2: 