# Data Cleaning

Data cleaning is a part of the process on a data science project.

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.
When you clean your data, all outdated or incorrect information is gone – leaving you with the highest quality information for your analysis and model building.

4 points

In [1]:

```
# import pandas and numpy with alias pd and np respectively
```

Create a dataframe, d1 = pd.DataFrame( {'Temperature' : [1, np.nan, 3, 2, 3] ,'Humidity' : [22, np.nan, 2 , np.nan, 20 ] })

In [2]:

```
#create d1
```

print the dataframe d1

In [3]:

```
#print d1
```

Out[3]:

| | Temperature | Humidity |
|---|---|---|
| **0** | 1.0 | 22.0 |
| **1** | NaN | NaN |
| **2** | 3.0 | 2.0 |
| **3** | 2.0 | NaN |
| **4** | 3.0 | 20.0 |

Find whether the given dataframe contain any missing values?

In [4]:

```
#check for null
```

Out[4]:

| | Temperature | Humidity |
|---|---|---|
| **0** | False | False |
| **1** | True | True |
| **2** | False | False |
| **3** | False | True |
| **4** | False | False |

How many missing values does each column have?

In [5]:

```
#total null
```

Out[5]:

```
Temperature    1
Humidity       2
dtype: int64
```

## Dealing with missing values

Refrence:-https://www.youtube.com/watch?v=xkRz6R0FIQ4 (https://www.youtube.com/watch?v=xkRz6R0FIQ4)

Now, we know we have missing values, the next thing that we need to work on, is how to deal with these missing values

- **Method 1: Delete the rows which contain missing values.**
  This method include dropping all the rows that have missing value in any column.

20 points

Use a suitable method to drop all the rows having missing values and save the change in d2 variable

In [6]:

```
d2=#drop nan
```

Print d2

In [7]:

```
#print nan
```

Out[7]:

| | Temperature | Humidity |
|---|---|---|
| **0** | 1.0 | 22.0 |
| **2** | 3.0 | 2.0 |
| **4** | 3.0 | 20.0 |

Hey Remember : droping rows with nan is one of the method to deal with missing values. But you have to decide if you need to go for this method by checking percentage of nan present in the dataframe.

If a column is having more than 60% of nan values then its better to remove such variables altogether if business permits

- ## Method 2: Replacing missing values
  Sometimes rather than dropping NA values, you'd rather replace them with a valid value. This value might be a single number like zero, or it might be some sort of imputation or interpolation

30 points

Impute the missing values with constant number of your choice

In [8]:

```
# The below output has imputed missing  values with 100
```

Out[8]:

| | Temperature | Humidity |
|---|---|---|
| **0** | 1.0 | 22.0 |
| **1** | 100.0 | 100.0 |
| **2** | 3.0 | 2.0 |
| **3** | 2.0 | 100.0 |
| **4** | 3.0 | 20.0 |

Do you think its a good way to treat Nan values? What if such constant values are not suitable for our further analysis? Try to give your thoughts on this.

In [ ]:

```

```

Impute the missing values with mean

In [9]:

```
#  imputing mean
```

Out[9]:

| | Temperature | Humidity |
|---|---|---|
| **0** | 1.00 | 22.000000 |
| **1** | 2.25 | 14.666667 |
| **2** | 3.00 | 2.000000 |
| **3** | 2.00 | 14.666667 |
| **4** | 3.00 | 20.000000 |

Impute the missing values with median

In [10]:

```
#median imputing
```

Out[10]:

| | Temperature | Humidity |
|---|---|---|
| **0** | 1.0 | 22.0 |
| **1** | 2.5 | 20.0 |
| **2** | 3.0 | 2.0 |
| **3** | 2.0 | 20.0 |
| **4** | 3.0 | 20.0 |

Replacing with the mean, mode or median approximations are a statistical approach of handling the missing values.

Another Fun fact:

```
This is an approximation which can add variance to the data set. But the loss of t
he data can be negated by this method which yields better results compared to remo
val of rows and columns.
```

Impute Nan with forward fill

In [11]:

```
#forward fill
```

Out[11]:

|   | Temperature | Humidity |
|---|---|---|
| **0** | 1.0 | 22.0 |
| **1** | 1.0 | 22.0 |
| **2** | 3.0 | 2.0 |
| **3** | 2.0 | 2.0 |
| **4** | 3.0 | 20.0 |

Impute Nan with backward fill

In [12]:

```
#backward fill
```

Out[12]:

|   | Temperature | Humidity |
|---|---|---|
| **0** | 1.0 | 22.0 |
| **1** | 3.0 | 2.0 |
| **2** | 3.0 | 2.0 |
| **3** | 2.0 | 20.0 |
| **4** | 3.0 | 20.0 |

Hey a fun fact here, as sweet as a cookie:

- ffill/pad/bfill are good imputation method if our data is of time series. This would keep the trend unaffected for our analysis.

Impute nan using interpolation method

In [13]:

```
#interpolate
```

Out[13]:

| | Temperature | Humidity |
|---|---|---|
| **0** | 1.0 | 22.0 |
| **1** | 2.0 | 12.0 |
| **2** | 3.0 | 2.0 |
| **3** | 2.0 | 11.0 |
| **4** | 3.0 | 20.0 |

You lucky champ! you got to know another amazing fact:

- Interpolation method by default is linear in nature. It is an imputation technique that assumes a linear relationship between data points and utilises non-missing values from adjacent data points to compute a value for a missing data point.

You can explore other techniques involved in interplolation method, which might be usefull for your project.

Perform KNN imputation

Reference: https://link.medium.com/mV3023if8gb (https://link.medium.com/mV3023if8gb)

In [14]:

```
# Hint: Import KNNImputer and impute it on d1. Also note: Use n_neighbors=2
```

Out[14]:

| | Temperature | Humidity |
|---|---|---|
| **0** | 1.00 | 22.000000 |
| **1** | 2.25 | 14.666667 |
| **2** | 3.00 | 2.000000 |
| **3** | 2.00 | 12.000000 |
| **4** | 3.00 | 20.000000 |

Point to ponder: KNN is an algorithm that is useful for matching a point with its closest k neighbors in a multi-dimensional space.

Do you think scaling is required to implement this method?. Yes you are right the answer is YES. Can you comment below why normalized data is required, so that we understand your logic on this.

It requires us to normalize our data. Otherwise, the different scales of our data will lead the KNN Imputer to generate biased replacements for the missing values.

## Dropping Irrelevant Columns

Reference:-https://www.youtube.com/watch?v=cRurWEfmxC0 (https://www.youtube.com/watch?v=cRurWEfmxC0)

5 points

Create a dataframe df = pd.DataFrame(np.random.randint(0,100,size=(100, 5)), columns=list('ABCDE'))

In [15]:

```
np.random.seed(10)

df = #define df
```

print df

In [16]:

```
print df
```

Out[16]:

```
      A   B   C   D   E
 0    9  15  64  28  89
 1   93  29   8  73   0
 2   40  36  16  11  54
 3   88  62  33  72  78
 4   49  51  54  77  69
...  ...  ... ... ... ...
95    3  50  59  34  21
96   16  18  61  54  60
97   21  87  83  71  16
98   67  38  27  96  87
99   98  89  16  82  19
```

100 rows × 5 columns

Note: Since all the rows are having random numbers, your dataframe observations might be different than the output given above

Suppose for our analysis our project do not require column E. So you need to remove this column. update this new change using inplace parameter

In [17]:

```
#drop E
```

Check if column **E** is removed by printing head of df

In [18]:

```
#df head
```

Out[18]:

| | A | B | C | D |
|---|---|---|---|---|
| **0** | 9 | 15 | 64 | 28 |
| **1** | 93 | 29 | 8 | 73 |
| **2** | 40 | 36 | 16 | 11 |
| **3** | 88 | 62 | 33 | 72 |
| **4** | 49 | 51 | 54 | 77 |

# Ensure requirements as per domain

Reference:-https://www.youtube.com/watch?v=eM7p3MVLOZ8 (https://www.youtube.com/watch?v=eM7p3MVLOZ8)

10 points

Shallow copy the dataframe df in variable df2 and print df2 head

In [19]:

```
df2=#

#df2 head
```

Out[19]:

| | A | B | C | D |
|---|---|---|---|---|
| **0** | 9 | 15 | 64 | 28 |
| **1** | 93 | 29 | 8 | 73 |
| **2** | 40 | 36 | 16 | 11 |
| **3** | 88 | 62 | 33 | 72 |
| **4** | 49 | 51 | 54 | 77 |

Suppose your domain expert says to filter column B with even numbers to do correct analysis. Implement the same below and update the change in varaible df2.

In [20]:

```
df2=#
```

print updated head of *df2*

In [21]:

```
# df2 head
```

Out[21]:

|  | A | B | C | D |
|---|---|---|---|---|
| **2** | 40 | 36 | 16 | 11 |
| **3** | 88 | 62 | 33 | 72 |
| **6** | 30 | 30 | 89 | 12 |
| **9** | 11 | 28 | 74 | 88 |
| **10** | 15 | 18 | 80 | 71 |

# Creating a sensible index values

Reference:-https://www.youtube.com/watch?v=FQ4IB5mZOmQ (https://www.youtube.com/watch?v=FQ4IB5mZOmQ)

Oops. The index in this dataframe doesn't make sense. please correct index in a sequential manner starting from 1. Save the updates in df2

In [22]:

```
#set proper index
```

print df2 head again

In [23]:

```
# df2 head
```

Out[23]:

|  | A | B | C | D |
|---|---|---|---|---|
| **1** | 40 | 36 | 16 | 11 |
| **2** | 88 | 62 | 33 | 72 |
| **3** | 30 | 30 | 89 | 12 |
| **4** | 11 | 28 | 74 | 88 |
| **5** | 15 | 18 | 80 | 71 |

# Renaming column names to meaningful names.

Reference:- https://www.youtube.com/watch?v=YnggjqAfZCU (https://www.youtube.com/watch?v=YnggjqAfZCU)

2 points

Now df2 columns represents marks of the adventurous 'Anand', the brave 'Barkha', the compassionate 'Chandu' and the dashing 'Daniel'. Rename the columns with their name inplace of their first letter of their name.

In [24]:

```
#column renaming
```

print df2 tail with updated column names

In [25]:

```
# df2 head
```

Out[25]:

| | Anand | Barkha | Chandu | Daniel |
|---|---|---|---|---|
| **44** | 1 | 82 | 34 | 11 |
| **45** | 74 | 36 | 6 | 63 |
| **46** | 3 | 50 | 59 | 34 |
| **47** | 16 | 18 | 61 | 54 |
| **48** | 67 | 38 | 27 | 96 |

Yeah! now the data looks pretty meaningful to study

## Treating Duplicate Data

Reference: https://www.youtube.com/watch?v=bFVMR1qfzXo (https://www.youtube.com/watch?v=bFVMR1qfzXo)

20 points

Make another dataframe df3 by deep copying df2.

In [26]:

```
df3=#
```

Make another column in df3 with name 'dummy' having 0 as values throughout the rows.

In [27]:

```
#assign dummy column full of zero value
```

In [28]:

```
# print head of df, df2 and df3
```

```
    A   B   C   D
0   9  15  64  28
1  93  29   8  73
2  40  36  16  11
3  88  62  33  72
4  49  51  54  77
   Anand  Barkha  Chandu  Daniel
1     40      36      16      11
2     88      62      33      72
3     30      30      89      12
4     11      28      74      88
5     15      18      80      71
   Anand  Barkha  Chandu  Daniel  dummy
1     40      36      16      11      0
2     88      62      33      72      0
3     30      30      89      12      0
4     11      28      74      88      0
5     15      18      80      71      0
```

Hey buddy! Don't you think, there is some difference between copy operation used for creating df2 and df3.

If you think Yes, Then please comment below the difference

In [29]:

```
# comment
```

print tail of df3

In [30]:

```
# df3 tail
```

Out[30]:

| | Anand | Barkha | Chandu | Daniel | dummy |
|---|---|---|---|---|---|
| **44** | 1 | 82 | 34 | 11 | 0 |
| **45** | 74 | 36 | 6 | 63 | 0 |
| **46** | 3 | 50 | 59 | 34 | 0 |
| **47** | 16 | 18 | 61 | 54 | 0 |
| **48** | 67 | 38 | 27 | 96 | 0 |

make an array name 'ListB' with values of column 'Barkha'

In [31]:

```
ListB=
```

Print ListB

In [32]:

```
#print ListB
```

Out[32]:

```
array([36, 62, 30, 28, 18, 50, 88, 50, 80, 66, 96, 30,  4, 30,  2, 42, 94,
       18, 44, 68, 58, 48, 70, 22, 36, 32, 32, 96, 30, 86,  0, 76, 88, 64,
       52, 46, 20, 66, 56,  8, 68, 50, 28, 82, 36, 50, 18, 38])
```

Assign this array values as another column in df3 with name 'Anonymous'

In [33]:

```
#create Anonymous column
```

Create a dataframe 'ListA' with values of row index 3, 10 and 40

In [34]:

```
ListA=#
```

print ListA

In [35]:

```
# print ListA
```

Out[35]:

| | Anand | Barkha | Chandu | Daniel | dummy | Anonymous |
|---|---|---|---|---|---|---|
| **3** | 30 | 30 | 89 | 12 | 0 | 30 |
| **10** | 96 | 66 | 67 | 62 | 0 | 66 |
| **40** | 74 | 8 | 92 | 32 | 0 | 8 |

Concat ListA to df3 ignoring the index values of ListA so that we can maintain the sequential index value thoughout the dataframe.

In [36]:

```
df3=#
```

print head of df3

In [37]:

```
# df3 head
```

Out[37]:

| | Anand | Barkha | Chandu | Daniel | dummy | Anonymous |
|---|---|---|---|---|---|---|
| **0** | 40 | 36 | 16 | 11 | 0 | 36 |
| **1** | 88 | 62 | 33 | 72 | 0 | 62 |
| **2** | 30 | 30 | 89 | 12 | 0 | 30 |
| **3** | 11 | 28 | 74 | 88 | 0 | 28 |
| **4** | 15 | 18 | 80 | 71 | 0 | 18 |

Check if there is any duplicate rows present in the dataframe df3

In [38]:

```
#check duplicate

```

Out[38]:

| | Anand | Barkha | Chandu | Daniel | dummy | Anonymous |
|---|---|---|---|---|---|---|
| **48** | 30 | 30 | 89 | 12 | 0 | 30 |
| **49** | 96 | 66 | 67 | 62 | 0 | 66 |
| **50** | 74 | 8 | 92 | 32 | 0 | 8 |

By above output it seems we do have duplicated rows in our dataset

Drop duplicated rows using pandas function keeping first values of such duplicated observations

In [39]:

```
#drop duplicate
```

Check again if we have any duplicate row values present

In [40]:

```
#check duplicate
```

Out[40]:

| Anand | Barkha | Chandu | Daniel | dummy | Anonymous |
|---|---|---|---|---|---|

Yipeee!! Did you notice the dataframe is free from any duplicate rows now.

Drop any duplicate columns present in the dataframe df

In [41]:

```
df3=#

#print df3
df3
```

Out[41]:

| | Anand | Barkha | Chandu | Daniel | dummy |
|---|---|---|---|---|---|
| **0** | 40 | 36 | 16 | 11 | 0 |
| **1** | 88 | 62 | 33 | 72 | 0 |
| **2** | 30 | 30 | 89 | 12 | 0 |
| **3** | 11 | 28 | 74 | 88 | 0 |
| **4** | 15 | 18 | 80 | 71 | 0 |
| **5** | 88 | 50 | 54 | 34 | 0 |
| **6** | 77 | 88 | 15 | 6 | 0 |
| **7** | 97 | 50 | 45 | 40 | 0 |
| **8** | 81 | 80 | 41 | 90 | 0 |
| **9** | 96 | 66 | 67 | 62 | 0 |
| **10** | 88 | 96 | 73 | 40 | 0 |
| **11** | 28 | 30 | 89 | 25 | 0 |
| **12** | 33 | 4 | 87 | 94 | 0 |
| **13** | 68 | 30 | 70 | 74 | 0 |
| **14** | 9 | 2 | 65 | 13 | 0 |
| **15** | 62 | 42 | 34 | 40 | 0 |
| **16** | 32 | 94 | 86 | 58 | 0 |
| **17** | 45 | 18 | 50 | 44 | 0 |
| **18** | 6 | 44 | 9 | 50 | 0 |
| **19** | 44 | 68 | 14 | 4 | 0 |
| **20** | 39 | 58 | 81 | 39 | 0 |
| **21** | 69 | 48 | 60 | 58 | 0 |
| **22** | 5 | 70 | 61 | 16 | 0 |
| **23** | 4 | 22 | 19 | 85 | 0 |
| **24** | 10 | 36 | 0 | 30 | 0 |
| **25** | 85 | 32 | 81 | 13 | 0 |
| **26** | 31 | 32 | 96 | 38 | 0 |
| **27** | 0 | 96 | 25 | 63 | 0 |
| **28** | 2 | 30 | 17 | 59 | 0 |
| **29** | 63 | 86 | 29 | 71 | 0 |
| **30** | 19 | 0 | 7 | 93 | 0 |
| **31** | 58 | 76 | 10 | 79 | 0 |

| | Anand | Barkha | Chandu | Daniel | dummy |
|---|---|---|---|---|---|
| **32** | 7 | 88 | 96 | 38 | 0 |
| **33** | 27 | 64 | 49 | 90 | 0 |
| **34** | 27 | 52 | 90 | 57 | 0 |
| **35** | 99 | 46 | 46 | 45 | 0 |
| **36** | 84 | 20 | 49 | 33 | 0 |
| **37** | 77 | 66 | 37 | 31 | 0 |
| **38** | 82 | 56 | 41 | 90 | 0 |
| **39** | 74 | 8 | 92 | 32 | 0 |
| **40** | 87 | 68 | 86 | 14 | 0 |
| **41** | 50 | 50 | 85 | 14 | 0 |
| **42** | 1 | 28 | 58 | 10 | 0 |
| **43** | 1 | 82 | 34 | 11 | 0 |
| **44** | 74 | 36 | 6 | 63 | 0 |
| **45** | 3 | 50 | 59 | 34 | 0 |
| **46** | 16 | 18 | 61 | 54 | 0 |
| **47** | 67 | 38 | 27 | 96 | 0 |

Did you notice which Column is dropped? I am sure you noticed it. Name that column below

In [42]:

```
#Column:_____________
```

## Treating constant column values

2 points

Check unique values in each columns

In [43]:

```
# df3 unique value
```

Out[43]:

```
Anand     42
Barkha    32
Chandu    40
Daniel    36
dummy      1
dtype: int64
```

By above output which column has only 1 value as unique throught the rows? Yeah! you are right, its dummy column. So lets drop it

Drop dummy column as it has constant values which will not give us any information and save the changes to df3 using inplace parameter

In [44]:

```
# drop dummy
```

print final obtained dataframe df3

In [45]:

```
# print df3
```

Out[45]:

| | Anand | Barkha | Chandu | Daniel |
|---|---|---|---|---|
| **0** | 40 | 36 | 16 | 11 |
| **1** | 88 | 62 | 33 | 72 |
| **2** | 30 | 30 | 89 | 12 |
| **3** | 11 | 28 | 74 | 88 |
| **4** | 15 | 18 | 80 | 71 |
| **5** | 88 | 50 | 54 | 34 |
| **6** | 77 | 88 | 15 | 6 |
| **7** | 97 | 50 | 45 | 40 |
| **8** | 81 | 80 | 41 | 90 |
| **9** | 96 | 66 | 67 | 62 |
| **10** | 88 | 96 | 73 | 40 |
| **11** | 28 | 30 | 89 | 25 |
| **12** | 33 | 4 | 87 | 94 |
| **13** | 68 | 30 | 70 | 74 |
| **14** | 9 | 2 | 65 | 13 |
| **15** | 62 | 42 | 34 | 40 |
| **16** | 32 | 94 | 86 | 58 |
| **17** | 45 | 18 | 50 | 44 |
| **18** | 6 | 44 | 9 | 50 |
| **19** | 44 | 68 | 14 | 4 |
| **20** | 39 | 58 | 81 | 39 |
| **21** | 69 | 48 | 60 | 58 |
| **22** | 5 | 70 | 61 | 16 |
| **23** | 4 | 22 | 19 | 85 |
| **24** | 10 | 36 | 0 | 30 |
| **25** | 85 | 32 | 81 | 13 |
| **26** | 31 | 32 | 96 | 38 |
| **27** | 0 | 96 | 25 | 63 |
| **28** | 2 | 30 | 17 | 59 |
| **29** | 63 | 86 | 29 | 71 |
| **30** | 19 | 0 | 7 | 93 |
| **31** | 58 | 76 | 10 | 79 |
| **32** | 7 | 88 | 96 | 38 |
| **33** | 27 | 64 | 49 | 90 |

| | Anand | Barkha | Chandu | Daniel |
|---|---|---|---|---|
| **34** | 27 | 52 | 90 | 57 |
| **35** | 99 | 46 | 46 | 45 |
| **36** | 84 | 20 | 49 | 33 |
| **37** | 77 | 66 | 37 | 31 |
| **38** | 82 | 56 | 41 | 90 |
| **39** | 74 | 8 | 92 | 32 |
| **40** | 87 | 68 | 86 | 14 |
| **41** | 50 | 50 | 85 | 14 |
| **42** | 1 | 28 | 58 | 10 |
| **43** | 1 | 82 | 34 | 11 |
| **44** | 74 | 36 | 6 | 63 |
| **45** | 3 | 50 | 59 | 34 |
| **46** | 16 | 18 | 61 | 54 |
| **47** | 67 | 38 | 27 | 96 |

# Iterating dataframes

25 points

Reference: https://www.youtube.com/watch?v=0nI3HTLPpZI (https://www.youtube.com/watch?v=0nI3HTLPpZI)
Let's look at three main ways to iterate over DataFrames.

1. iteritems()
2. iterrows()
3. itertuples()

We will also see time taken by these methods to print our dataframe.

**1. Iterating DataFrames with iteritems()**

Lets iterate over rows of df3 uisng iteritems.

In [46]:

```
import time
start = time.time()

#Use iteritems to iterate



print('Time taken(sec): ',(time.time()-start)*1000)
```

```
Anand
0     40
1     88
2     30
3     11
4     15
5     88
6     77
7     97
8     81
9     96
10    88
11    28
12    33
13    68
14     9
15    62
16    32
17    45
```

Did you notice buddy how iteritems are iterating over df3.

Along with ways each iterating function works, also keep tallying the time taken for all other lopps too!. This will be fun, lets check iterrows()

**2. Iterating DataFrames with iterrows()**

In [47]:

```
import time
start = time.time()
#Use iterrows to iterate

print('Time taken(sec): ',(time.time()-start)*1000)
```

```
0
Anand      40
Barkha     36
Chandu     16
Daniel     11
Name: 0, dtype: int64
1
Anand      88
Barkha     62
Chandu     33
Daniel     72
Name: 1, dtype: int64
2
Anand      30
Barkha     30
Chandu     89
Daniel     12
Name: 2, dtype: int64
3
```

### 3. Iterating DataFrames with itertuples()

In [48]:

```
#iterate df3 using itertuples
import time
start = time.time()

#Use itertuples to iterate


print('Time taken(sec): ',(time.time()-start)*1000)
```

```
Pandas(Index=0, Anand=40, Barkha=36, Chandu=16, Daniel=11)
Pandas(Index=1, Anand=88, Barkha=62, Chandu=33, Daniel=72)
Pandas(Index=2, Anand=30, Barkha=30, Chandu=89, Daniel=12)
Pandas(Index=3, Anand=11, Barkha=28, Chandu=74, Daniel=88)
Pandas(Index=4, Anand=15, Barkha=18, Chandu=80, Daniel=71)
Pandas(Index=5, Anand=88, Barkha=50, Chandu=54, Daniel=34)
Pandas(Index=6, Anand=77, Barkha=88, Chandu=15, Daniel=6)
Pandas(Index=7, Anand=97, Barkha=50, Chandu=45, Daniel=40)
Pandas(Index=8, Anand=81, Barkha=80, Chandu=41, Daniel=90)
Pandas(Index=9, Anand=96, Barkha=66, Chandu=67, Daniel=62)
Pandas(Index=10, Anand=88, Barkha=96, Chandu=73, Daniel=40)
Pandas(Index=11, Anand=28, Barkha=30, Chandu=89, Daniel=25)
Pandas(Index=12, Anand=33, Barkha=4, Chandu=87, Daniel=94)
Pandas(Index=13, Anand=68, Barkha=30, Chandu=70, Daniel=74)
Pandas(Index=14, Anand=9, Barkha=2, Chandu=65, Daniel=13)
Pandas(Index=15, Anand=62, Barkha=42, Chandu=34, Daniel=40)
Pandas(Index=16, Anand=32, Barkha=94, Chandu=86, Daniel=58)
Pandas(Index=17, Anand=45, Barkha=18, Chandu=50, Daniel=44)
Pandas(Index=18, Anand=6, Barkha=44, Chandu=9, Daniel=50)
Pandas(Index=19, Anand=44, Barkha=68, Chandu=14, Daniel=4)
Pandas(Index=20, Anand=39, Barkha=58, Chandu=81, Daniel=39)
Pandas(Index=21, Anand=69, Barkha=48, Chandu=60, Daniel=58)
Pandas(Index=22, Anand=5, Barkha=70, Chandu=61, Daniel=16)
Pandas(Index=23, Anand=4, Barkha=22, Chandu=19, Daniel=85)
Pandas(Index=24, Anand=10, Barkha=36, Chandu=0, Daniel=30)
Pandas(Index=25, Anand=85, Barkha=32, Chandu=81, Daniel=13)
Pandas(Index=26, Anand=31, Barkha=32, Chandu=96, Daniel=38)
Pandas(Index=27, Anand=0, Barkha=96, Chandu=25, Daniel=63)
Pandas(Index=28, Anand=2, Barkha=30, Chandu=17, Daniel=59)
Pandas(Index=29, Anand=63, Barkha=86, Chandu=29, Daniel=71)
Pandas(Index=30, Anand=19, Barkha=0, Chandu=7, Daniel=93)
Pandas(Index=31, Anand=58, Barkha=76, Chandu=10, Daniel=79)
Pandas(Index=32, Anand=7, Barkha=88, Chandu=96, Daniel=38)
Pandas(Index=33, Anand=27, Barkha=64, Chandu=49, Daniel=90)
Pandas(Index=34, Anand=27, Barkha=52, Chandu=90, Daniel=57)
Pandas(Index=35, Anand=99, Barkha=46, Chandu=46, Daniel=45)
Pandas(Index=36, Anand=84, Barkha=20, Chandu=49, Daniel=33)
Pandas(Index=37, Anand=77, Barkha=66, Chandu=37, Daniel=31)
Pandas(Index=38, Anand=82, Barkha=56, Chandu=41, Daniel=90)
Pandas(Index=39, Anand=74, Barkha=8, Chandu=92, Daniel=32)
Pandas(Index=40, Anand=87, Barkha=68, Chandu=86, Daniel=14)
Pandas(Index=41, Anand=50, Barkha=50, Chandu=85, Daniel=14)
Pandas(Index=42, Anand=1, Barkha=28, Chandu=58, Daniel=10)
Pandas(Index=43, Anand=1, Barkha=82, Chandu=34, Daniel=11)
Pandas(Index=44, Anand=74, Barkha=36, Chandu=6, Daniel=63)
Pandas(Index=45, Anand=3, Barkha=50, Chandu=59, Daniel=34)
Pandas(Index=46, Anand=16, Barkha=18, Chandu=61, Daniel=54)
Pandas(Index=47, Anand=67, Barkha=38, Chandu=27, Daniel=96)
Time taken(sec):  12.319803237915039
```

Hey buddy! so as you have seen every method works differently

```
iteritems(): Helps to iterate over each element of the set, column-wise.
iterrows(): Each element of the set, row-wise.
itertuple(): Each row and form a tuple out of them.
```

But if you ask for speed. The most best perfromance is given by itertuples compared to other two iterating methods. So if anytime you need to save your computation time on iteration of dataframes you can go for itertuples. Was'nt it fun?:)

# Regular Expression

15 points

Reference: https://www.youtube.com/watch?v=zTTQ8FE60j8 (https://www.youtube.com/watch?v=zTTQ8FE60j8)

Reference: https://www.youtube.com/watch?v=h6E1PiTXnVI (https://www.youtube.com/watch?v=h6E1PiTXnVI)

Reference doc: https://www.guru99.com/python-regular-expressions-complete-tutorial.html (https://www.guru99.com/python-regular-expressions-complete-tutorial.html)

Python has a module named re to work with RegEx

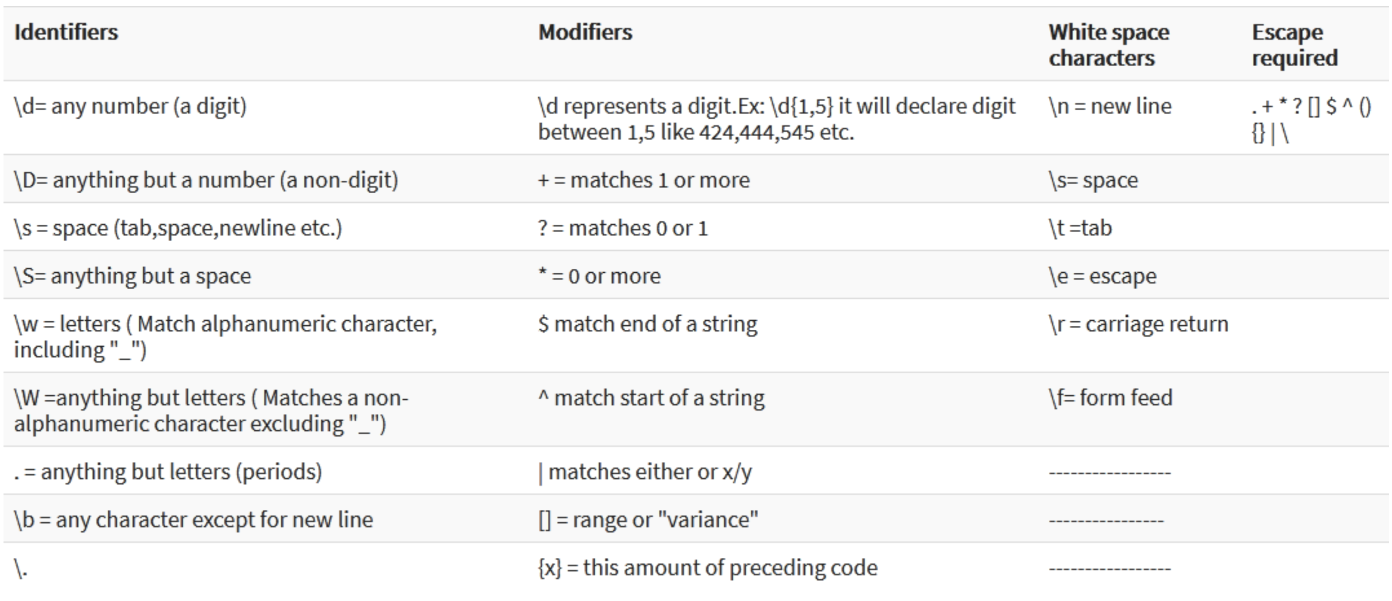
## !Are you ready to try regex on dataframes?

*So here we go.!*

We are gonna try out following awesome re module functions

1. findall
2. search
3. sub
4. split

If you want you can also refer the below regular expression syntax.



| Identifiers | Modifiers | White space characters | Escape required |
|---|---|---|---|
| \d= any number (a digit) | \d represents a digit.Ex: \d{1,5} it will declare digit between 1,5 like 424,444,545 etc. | \n = new line | . + * ? [] $ ^ () {} \| \ |
| \D= anything but a number (a non-digit) | + = matches 1 or more | \s= space | |
| \s = space (tab,space,newline etc.) | ? = matches 0 or 1 | \t =tab | |
| \S= anything but a space | * = 0 or more | \e = escape | |
| \w = letters ( Match alphanumeric character, including "_") | $ match end of a string | \r = carriage return | |
| \W =anything but letters ( Matches a non-alphanumeric character excluding "_") | ^ match start of a string | \f= form feed | |
| . = anything but letters (periods) | \| matches either or x/y | ----------------- | |
| \b = any character except for new line | [] = range or "variance" | ---------------- | |
| \. | {x} = this amount of preceding code | ----------------- | |

Hey future data scientists! we will now use regex on dataframes for data cleaning.

Who doesn't know Trump?. Lets dowload this interesting dataset of Trump insult tweets :https://www.kaggle.com/ayushggarg/all-trumps-twitter-insults-20152021/download (https://www.kaggle.com/ayushggarg/all-trumps-twitter-insults-20152021/download)

On this dataset we will learn how to use regex for data cleaning. By the way it will be also very usefull for feature engineering too!.

In [49]:

```
#load dataset
tweet_data=#
```

In [50]:

```
#import re module


#print head of tweet_data
```

Out[50]:

| | Unnamed: 0 | date | target | insult | tweet |
|---|---|---|---|---|---|
| **0** | 1 | 2014-10-09 | thomas-frieden | fool | Can you believe this fool, Dr. Thomas Frieden ... |
| **1** | 2 | 2014-10-09 | thomas-frieden | DOPE | Can you believe this fool, Dr. Thomas Frieden ... |
| **2** | 3 | 2015-06-16 | politicians | all talk and no action | Big time in U.S. today - MAKE AMERICA GREAT AG... |
| **3** | 4 | 2015-06-24 | ben-cardin | It's politicians like Cardin that have destroy... | Politician @SenatorCardin didn't like that I s... |
| **4** | 5 | 2015-06-24 | neil-young | total hypocrite | For the nonbeliever, here is a photo of @Neily... |

Lets do some analysis using regex on this dataset

Before we go ahead, do you remember apply function? because you will have to require apply function to impliment regex methods.

You can refer this video so that you revisit how apply function works: https://www.youtube.com/watch?v=7HN-4Df8ZpA (https://www.youtube.com/watch?v=7HN-4Df8ZpA)

**1. findall()**

Make another column 'year' with year in each row using regex on date column.

In [51]:

```
# create a function which takes date as parameter and applies regex on it


#use apply function on tweet_data to use above function in order to make year column


#print tweet_data head

```

Out[51]:

| | Unnamed: 0 | date | target | insult | tweet | year |
|---|---|---|---|---|---|---|
| **0** | 1 | 2014-10-09 | thomas-frieden | fool | Can you believe this fool, Dr. Thomas Frieden ... | 2014 |
| **1** | 2 | 2014-10-09 | thomas-frieden | DOPE | Can you believe this fool, Dr. Thomas Frieden ... | 2014 |
| **2** | 3 | 2015-06-16 | politicians | all talk and no action | Big time in U.S. today - MAKE AMERICA GREAT AG... | 2015 |
| **3** | 4 | 2015-06-24 | ben-cardin | It's politicians like Cardin that have destroy... | Politician @SenatorCardin didn't like that I s... | 2015 |
| **4** | 5 | 2015-06-24 | neil-young | total hypocrite | For the nonbeliever, here is a photo of @Neily... | 2015 |

lets filter year from 2020-2021 which was the election time in USA.

### 2. search

We will use regex search for this

In [52]:

```
# create a function which takes year as parameter and applies regex on it


#use apply function on tweet_data to use above function in order to search tweets of 2020-2


#tweet_data head
```

Out[52]:

| | Unnamed: 0 | date | target | insult | tweet | year |
|---|---|---|---|---|---|---|
| **0** | 1 | 2014-10-09 | thomas-frieden | fool | Can you believe this fool, Dr. Thomas Frieden ... | None |
| **1** | 2 | 2014-10-09 | thomas-frieden | DOPE | Can you believe this fool, Dr. Thomas Frieden ... | None |
| **2** | 3 | 2015-06-16 | politicians | all talk and no action | Big time in U.S. today - MAKE AMERICA GREAT AG... | None |
| **3** | 4 | 2015-06-24 | ben-cardin | It's politicians like Cardin that have destroy... | Politician @SenatorCardin didn't like that I s... | None |
| **4** | 5 | 2015-06-24 | neil-young | total hypocrite | For the nonbeliever, here is a photo of @Neily... | None |

You can also do the same thing using regex match function to do this which is vailable in pandas

Reference: https://www.geeksforgeeks.org/python-pandas-series-str-match/ (https://www.geeksforgeeks.org/python-pandas-series-str-match/)

In [53]:

```
# apply pandas str.match() function
```

Out[53]:

| | Unnamed: 0 | date | target | insult | tweet | year |
|---|---|---|---|---|---|---|
| **7621** | 7622 | 2020-01-01 | impeachment-inquiry | The greatest Witch Hunt in U.S. history | Thank you Steve. The greatest Witch Hunt in U.... | 2020 |
| **7622** | 7623 | 2020-01-02 | impeachment-inquiry | Witch Hunt is sputtering badly, but still goin... | A lot of very good people were taken down by a... | 2020 |
| **7623** | 7624 | 2020-01-04 | impeachment-inquiry | this ridiculous Impeachment Lite Hoax | As hard as I work, & as successful as our Coun... | 2020 |
| **7624** | 7625 | 2020-01-06 | los-angeles | poorly run | ....If however, the city or state in question ... | 2020 |
| **7625** | 7626 | 2020-01-06 | impeachment-inquiry | The great Scam continues | “The reason they are not sending the Articles ... | 2020 |
| **...** | ... | ... | ... | ... | ... | ... |
| **10355** | 10356 | 2021-01-06 | 2020-election | Many States want to decertify the mistake they... | If Vice President @Mike_Pence comes through fo... | 2021 |
| **10356** | 10357 | 2021-01-06 | 2020-election | based on irregularities and fraud, plus corrup... | States want to correct their votes, which they... | 2021 |
| **10357** | 10358 | 2021-01-06 | 2020-election | Our Election Process is worse than that of thi... | They just happened to find 50,000 ballots late... | 2021 |
| **10358** | 10359 | 2021-01-06 | 2020-election | a FRAUD | The States want to redo their votes. They foun... | 2021 |
| **10359** | 10360 | 2021-01-06 | chuck-todd | Sleepy Eyes, Sad to watch! | Sleepy Eyes Chuck Todd is so happy with the fa... | 2021 |

2739 rows × 6 columns

cool right!

You got some null values after applying above function. Lets drop them using dropna function. Also drop 'Unnamed: 0' column as it does not give any information.

In [54]:

```
#drop na and Unnamed: 0 column
```

**3. sub() Function**

Now you have filtered the dataset with 2739 rows. Let's remove all @ from tweet column suing re sub() function.

In [55]:

```
# create a function which takes tweet as parameter and applies regex on it


#use apply function on tweet_data to use above function in order to remove @ from tweets

#tweet_data head
```

Out[55]:

| | date | target | insult | tweet | year |
|---|---|---|---|---|---|
| **7621** | 2020-01-01 | impeachment-inquiry | The greatest Witch Hunt in U.S. history | Thank you Steve. The greatest Witch Hunt in U.... | 2020 |
| **7622** | 2020-01-02 | impeachment-inquiry | Witch Hunt is sputtering badly, but still goin... | A lot of very good people were taken down by a... | 2020 |
| **7623** | 2020-01-04 | impeachment-inquiry | this ridiculous Impeachment Lite Hoax | As hard as I work, & as successful as our Coun... | 2020 |
| **7624** | 2020-01-06 | los-angeles | poorly run | ....If however, the city or state in question ... | 2020 |
| **7625** | 2020-01-06 | impeachment-inquiry | The great Scam continues | “The reason they are not sending the Articles ... | 2020 |
| **...** | ... | ... | ... | ... | ... |
| **10355** | 2021-01-06 | 2020-election | Many States want to decertify the mistake they... | If Vice President Mike_Pence comes through for... | 2021 |
| **10356** | 2021-01-06 | 2020-election | based on irregularities and fraud, plus corrup... | States want to correct their votes, which they... | 2021 |
| **10357** | 2021-01-06 | 2020-election | Our Election Process is worse than that of thi... | They just happened to find 50,000 ballots late... | 2021 |
| **10358** | 2021-01-06 | 2020-election | a FRAUD | The States want to redo their votes. They foun... | 2021 |
| **10359** | 2021-01-06 | chuck-todd | Sleepy Eyes, Sad to watch! | Sleepy Eyes Chuck Todd is so happy with the fa... | 2021 |

2739 rows × 5 columns

You can also use the sub function just in one line using list comprehension. Can you try doing it below?

In [56]:

```
# sub() suing list comprehension
```

**4. split() Function**

Lets now split the column target by making another column Name which has name before hyphen("-")

In [57]:

```
# create a function which takes target as parameter and applies regex on it


#use apply function on tweet_data to use above function in order to create Name column

#tweet_data head
```

Out[57]:

| | date | target | insult | tweet | year | Name |
|---|---|---|---|---|---|---|
| **7621** | 2020-01-01 | impeachment-inquiry | The greatest Witch Hunt in U.S. history | Thank you Steve. The greatest Witch Hunt in U.... | 2020 | impeachment |
| **7622** | 2020-01-02 | impeachment-inquiry | Witch Hunt is sputtering badly, but still goin... | A lot of very good people were taken down by a... | 2020 | impeachment |
| **7623** | 2020-01-04 | impeachment-inquiry | this ridiculous Impeachment Lite Hoax | As hard as I work, & as successful as our Coun... | 2020 | impeachment |
| **7624** | 2020-01-06 | los-angeles | poorly run | ....If however, the city or state in question ... | 2020 | los |
| **7625** | 2020-01-06 | impeachment-inquiry | The great Scam continues | “The reason they are not sending the Articles ... | 2020 | impeachment |

Now can you filter out Name which are specially targetted for trump? Lets do it below and check how many such tweets are there.

In [58]:

```
#filter Name which is euqals to trump
```

Out[58]:

| | date | target | insult | tweet | year | Name |
|---|---|---|---|---|---|---|
| **7859** | 2020-02-09 | trump-russia | THE WHOLE SCAM INVESTIGATION, THE MUELLER REPO... | FBI Director Christopher Wray just admitted th... | 2020 | trump |
| **7860** | 2020-02-09 | trump-russia | the biggest political crime in American Histor... | ....This is the biggest political crime in Ame... | 2020 | trump |
| **7861** | 2020-02-09 | trump-russia | THE PARTY IN POWER ILLEGALLY SPIED ON MY CAMPAIGN | ....This is the biggest political crime in Ame... | 2020 | trump |
| **7890** | 2020-02-12 | trump-russia | an investigation that was illegal | Who are the four prosecutors (Mueller people?)... | 2020 | trump |
| **7891** | 2020-02-12 | trump-russia | the Mueller Scam | Who are the four prosecutors (Mueller people?)... | 2020 | trump |
| **...** | ... | ... | ... | ... | ... | ... |
| **9658** | 2020-10-07 | trump-russia | Hoax Scandal | All Russia Hoax Scandal information was Declas... | 2020 | trump |
| **9659** | 2020-10-07 | trump-russia | the biggest political crime in the history of ... | All Russia Hoax Scandal information was Declas... | 2020 | trump |
| **10091** | 2020-11-22 | trump-russia | never ending Witch Hunt | Thanks Mark. It's all a continuation of the ne... | 2020 | trump |
| **10188** | 2020-12-09 | trump-critics | obnoxious | Germany has consistently been used by my obnox... | 2020 | trump |
| **10258** | 2020-12-18 | trump-russia | Russia Hoax | The Russia Hoax becomes an even bigger lie! ht... | 2020 | trump |

65 rows × 6 columns

So here you got total of 65 records which were tweeted on Donald Trump in the span of 2020-2021.

Well done buddy! You have learned how to apply regex on dataframes. Regex are mostly used for datasets which are having textual information.

Good job! Now our interesting trump insult tweet data is somewhat cleaned.

---

# C'mon cheers:) you have completed the 5th milestone challenge too.

---

# FeedBack

We hope you’ve enjoyed this course so far. We’re committed to help you use "AI for All" course to its full potential, so that you have a great learning experience. And that’s why we need your help in form of a feedback here.

Please fill this feedback form
https://docs.google.com/forms/d/e/1FAIpQLSfjBmH0yJSSA34IhSVx4h2eDMgOAeG4Dk-yHid__NMTk3Hq5g/viewform
(https://docs.google.com/forms/d/e/1FAIpQLSfjBmH0yJSSA34IhSVx4h2eDMgOAeG4Dk-yHid__NMTk3Hq5g/viewform)