It happens all the time: someone gives you data containing malformed strings, Python, lists and missing data. How do you tidy it up so you can get on with the analysis?

Take this monstrosity as the DataFrame to use in the following puzzles:

df = pd.DataFrame({'From_To': ['LoNDon_paris', 'MAdrid_miLAN', 'londON_StockhOlm',
'Budapest_PaRis', 'Brussels_londOn'],

'FlightNumber': [10045, np.nan, 10065, np.nan, 10085],

'RecentDelays': [[23, 47], [], [24, 43, 87], [13], [67, 32]],

'Airline': ['KLM(!)', ' (12)', '(British Airways.)',

'12. Air France', '"Swiss Air"']})

1. Some values in the the FlightNumber column are missing. These numbers are meant to increase by 10 with each row so 10055 and 10075 need to be put in place. Fill in these missing numbers and make the column an integer column (instead of a float column).
2. The From_To column would be better as two separate columns! Split each string on the underscore delimiter _ to give a new temporary DataFrame with the correct values. Assign the correct column names to this temporary DataFrame.
3. Notice how the capitalisation of the city names is all mixed up in this temporary DataFrame. Standardise the strings so that only the first letter is uppercase (e.g. "londON" should become "London".)
4. Delete the From_To column from df and attach the temporary DataFrame from the previous questions.
5. In the RecentDelays column, the values have been entered into the DataFrame as a list. We would like each first value in its own column, each second value in its own column, and so on. If there isn't an Nth value, the value should be NaN.

Expand the Series of lists into a DataFrame named delays, rename the columns delay_1, delay_2, etc. and replace the unwanted RecentDelays column in df with delays.

In [35]:
```
import numpy as np
import pandas as pd
```

In [36]:
```
df = pd.DataFrame({'From_To': ['LoNDon_paris', 'MAdrid_miLAN',  'londON_StockhOlm

'FlightNumber': [10045, np.nan, 10065, np.nan, 10085],

'RecentDelays': [[23, 47], [], [24, 43, 87], [13], [67, 32]],

'Airline': ['KLM(!)', '<Air France> (12)', '(British Airways. )',

'12. Air France', '"Swiss Air"']})
```

In [37]:
```
df.head()
```

Out[37]:

	From_To	FlightNumber	RecentDelays	Airline
0	LoNDon_paris	10045.0	[23, 47]	KLM(!)
1	MAdrid_miLAN	NaN	[]	<Air France> (12)
2	londON_StockhOlm	10065.0	[24, 43, 87]	(British Airways.)
3	Budapest_PaRis	NaN	[13]	12. Air France
4	Brussels_londOn	10085.0	[67, 32]	"Swiss Air"

1. Some values in the the FlightNumber column are missing. These numbers are meant to increase by 10 with each row so 10055 and 10075 need to be put in place. Fill in these missing numbers and make the column an integer column (instead of a float column)

In [38]:
```
new_df = df.interpolate()
new_df.head()
```

Out[38]:

	From_To	FlightNumber	RecentDelays	Airline
0	LoNDon_paris	10045.0	[23, 47]	KLM(!)
1	MAdrid_miLAN	10055.0	[]	<Air France> (12)
2	londON_StockhOlm	10065.0	[24, 43, 87]	(British Airways.)
3	Budapest_PaRis	10075.0	[13]	12. Air France
4	Brussels_londOn	10085.0	[67, 32]	"Swiss Air"

2. The From_To column would be better as two separate columns! Split each string on the underscore delimiter _ to give a new temporary DataFrame with the correct values. Assign the correct column names to this temporary DataFrame.

In [39]:
```
new_dfi = new_df["From_To"].str.split("_",expand=True)
new_dfi.head()
```

Out[39]:

	0	1
0	LoNDon	paris
1	MAdrid	miLAN
2	londON	StockhOlm
3	Budapest	PaRis
4	Brussels	londOn

In [40]:
```
new_df.drop(columns= "From_To",inplace = True)
new_df
```

...

In [41]:
```
new_df2  = new_dfi.rename(columns= { 0:'From' , 1 : "To"})
new_df2.head()
```

...

3. Notice how the capitalisation of the city names is all mixed up in this temporary DataFrame. Standardise the strings so that only the first letter is uppercase (e.g. "londON" should become "London".)

In [42]:
```
new_df2["From"] = new_df2.From.str.title()
new_df2["To"] = new_df2.To.str.title()
```

In [43]:
```
new_df2.head()
```

...

4. Delete the From_To column from df and attach the temporary DataFrame from the previous questions.

In [44]:
```
new_df3 = pd.merge(new_df , new_df2 , left_index=True,right_index=True)
new_df3.head()
```

...

5. In the RecentDelays column, the values have been entered into the DataFrame as a list. We would like each first value in its own column, each second value in its own column, and so on. If there isn't an Nth value, the value should be NaN.

Expand the Series of lists into a DataFrame named delays, rename the columns delay_1, delay_2, etc. and replace the unwanted RecentDelays column in df with delays.

In [45]:
```
new_df4 = df["RecentDelays"].astype(str) # Convert data to str
```

In [46]:
```
for i in range(len(new_df4)):
    if len(new_df4[i])>2:
        Update = new_df4[i][1:-1]
        new_df4[i]=new_df4[i][1:-1]
    else:#new_df4[i]=[]
        Update = new_df4[i][1:-1]
        new_df4[i]="NaN"
```

In [47]:
```
new_df4.fillna(value ="NaN")
```

Out[47]:
```
0        23, 47
1           NaN
2    24, 43, 87
3            13
4        67, 32
Name: RecentDelays, dtype: object
```

In [48]:
```
new_df5=new_df4.str.split(",",expand = True)
```

In [49]:
```
new  = new_df5.rename(columns= { 0:'delay_1' , 1 : "delay_2",2:"delay_3"})
```

In [50]:
```
Final = pd.merge(new_df3 , new , left_index=True,right_index=True)
```

In [51]:
```
Final.drop(columns= "RecentDelays",inplace = True)
```

In [52]:
```
Final_df = Final.fillna(value = "NaN")
```

In [53]:
```
Final_df
```

Out[53]:

	FlightNumber	Airline	From	To	delay_1	delay_2	delay_3
0	10045.0	KLM(!)	London	Paris	23	47	NaN
1	10055.0	<Air France> (12)	Madrid	Milan	NaN	NaN	NaN
2	10065.0	(British Airways.)	London	Stockholm	24	43	87
3	10075.0	12. Air France	Budapest	Paris	13	NaN	NaN
4	10085.0	"Swiss Air"	Brussels	London	67	32	NaN