## `import` Packages

In [1]:
```
import numpy as np
import pandas as pd
import random
```

## Creating dummy data set

In [2]:
```
real_values = np.random.rand(10, 3)
num_cate = np.array([random.choice([1, 2, 3]) for i in range(10)])
str_cate = np.array([random.choice(['short', 'tall', 'average']) for i in range(10)])
```

## Creating `df`

In [3]:
```
df = pd.DataFrame(data=real_values, columns=['col_1', 'col_2', 'col_3'])
```

In [4]:
```
df.head()
```

Out[4]:

| | col_1 | col_2 | col_3 |
|---|---|---|---|
| **0** | 0.902552 | 0.212317 | 0.208628 |
| **1** | 0.308503 | 0.959903 | 0.631784 |
| **2** | 0.459789 | 0.199742 | 0.176000 |
| **3** | 0.760787 | 0.167179 | 0.137287 |
| **4** | 0.199737 | 0.120213 | 0.145068 |

## Appending new columns

In [5]:
```
df['col_4'] = num_cate
df['col_5'] = str_cate
```

In [6]:
```
df.head()
```

Out[6]:

| | col_1 | col_2 | col_3 | col_4 | col_5 |
|---|---|---|---|---|---|
| **0** | 0.902552 | 0.212317 | 0.208628 | 2 | short |
| **1** | 0.308503 | 0.959903 | 0.631784 | 3 | tall |
| **2** | 0.459789 | 0.199742 | 0.176000 | 2 | tall |
| **3** | 0.760787 | 0.167179 | 0.137287 | 2 | average |
| **4** | 0.199737 | 0.120213 | 0.145068 | 2 | average |

In [7]:
```
df.dtypes
```

Out[7]:
```
col_1    float64
col_2    float64
col_3    float64
col_4      int32
col_5     object
dtype: object
```

In [8]:
```
def split_categories_numericals(dframe):
    cols = list(dframe.columns)
    num_cols = list(dframe._get_numeric_data().columns)
    cate_cols = list(set(cols) - set(num_cols))
    return cate_cols, num_cols
```

The above only works for categorical and numerical data values.

In [9]:
```
cate_cols, num_cols = split_categories_numericals(dframe=df)
```

## Columns that are categorical

In [10]:
```
cate_cols
```

Out[10]:
```
['col_5']
```

## Columns that are both numeric and float

In [11]:
```
num_cols
```

Out[11]:
```
['col_1', 'col_2', 'col_3', 'col_4']
```

## Question

- If I have a textual data (different from `col_5`), how can I know let `pandas` know that it is textual data?
- `._get_numeric_data()` is a method of `pandas` used to separate numerical and float columns from categorical data ( `col_5` )

In [12]:
```
# Imagine I have actual sentences in the data
# Just for example purpose, I am taking one sentence

text_data = ['hi hello how are you doing' for i in range(10)]
```

In [13]:
```
df['col_6'] = text_data
```

In [14]:
```
df.head()
```

Out[14]:

| | col_1 | col_2 | col_3 | col_4 | col_5 | col_6 |
|---|---|---|---|---|---|---|
| **0** | 0.902552 | 0.212317 | 0.208628 | 2 | short | hi hello how are you doing |
| **1** | 0.308503 | 0.959903 | 0.631784 | 3 | tall | hi hello how are you doing |
| **2** | 0.459789 | 0.199742 | 0.176000 | 2 | tall | hi hello how are you doing |
| **3** | 0.760787 | 0.167179 | 0.137287 | 2 | average | hi hello how are you doing |
| **4** | 0.199737 | 0.120213 | 0.145068 | 2 | average | hi hello how are you doing |

## Solution

In [15]:
```
def split_text_categories_numericals(dframe):
    cols = list(dframe.columns)

    num_cols = list(dframe._get_numeric_data().columns)
    cate_cols = list(set(cols) - set(num_cols))

    text_cols = []; category_cols = []
    for ccol in cate_cols:
        each_col_list = dframe[ccol].str.split(' ').to_list()
        col_val_len_arr = np.array(list(map(len, each_col_list)))

        if np.any(col_val_len_arr > 1):
            text_cols.append(ccol)
        else:
            category_cols.append(ccol)

    return num_cols, category_cols, text_cols
```

With a simple hack, we can now separate categorical, numerical, textual data.

In [16]:
```
num_cols, cate_cols, text_cols = split_text_categories_numericals(dframe=df)
```

In [17]:
```
num_cols
```

Out[17]:
```
['col_1', 'col_2', 'col_3', 'col_4']
```

In [18]:
```
cate_cols
```

Out[18]:
```
['col_5']
```

In [19]:
```
text_cols
```

Out[19]:
```
['col_6']
```

## Question

- Is there any hack to do it?
  - Yes, very well. But may not work most of the time, unless the data is cleaned and processed enough.

In [ ]:
```

```

**Thanks and Regards**