Hands-on Data Cleaning: Datasets

- File CSV tentang “Daftar Buku dari British Library”, nama ;le “`BL-Flickr- Images-Book.csv`”, link:
  - https://github.com/realpython/python-data-cleaning/blob/master/Datasets/BL-Flickr- Images-Book.csv
- File teks tentang “Kota lokasi Sekolah Tinggi di US”, nama ;le “`university_towns.txt`”, link: https://github.com/realpython/python-data-cleaning/blob/master/Datasets/university_towns.txt
- File CSV tentang “Partisipasi Semua Negara di Olimpiade Musim Dingin dan Musim
  - Panas”, nama ;le “`olympics.csv`”, link: https://github.com/realpython/python-data-cleaning/blob/master/Datasets/olympics.csv

Hands-on Data Cleaning: Import modul

Diasumsikan peserta sudah memahami library Pandas dan NumPy (lihat di modul sebelumnya) termasuk Pandas workshouse Series dan objek `DataFrame`

1. Import modul yang dibutuhkan

```
import pandas as pd
import numpy as np
```

Jika ingin melihat statistik dasar pada DataFrame di Pandas dengan fungsi .describe():

```
df.describe()
```

Hands-on Data Cleaning: Membuang (drop) Kolom

- Membuang Kolom pada `DataFrame`
- Sering ditemukan bbrp kategori data tidak terlalu berguna di dataset, misal untuk menganalisis IPK mahasiswa , data nama orangtua, alamat adalah data tidak penting
- Pandas menyediakan fungsi untuk membuang (drop) kolom-kolom yang tidak diinginkan dengan fungsi `drop()`.
  1. Buat DataFrame di luar ;le CSV . Dalam contoh berikut kita lewatkan path relatif ke `pd.read.csv`, yaitu seluruh dataset berada di nama folder Datasets di direktori kerja

Hands-on Data Cleaning: Membuang (drop) Kolom

```
df = pd.read_csv("C:/Users/Bayu/Documents/DTS 2021/Datasets/BL-Flickr-Images-Book.csv")
df.head()
```

	Identifier	Edition Statement	Place of Publication	Date of Publication	Publisher	Title	Author	Contributors	Corporate Author	Corporate Contributors	Former owner	Engraver	Issuance type	
0	206	NaN	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	FORBES, Walter.	NaN	NaN	NaN	NaN	monographic	http
1	216	NaN	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose - Baroness	NaN	NaN	NaN	NaN	monographic	http
2	218	NaN	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of "All for Gr...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose - Baroness	NaN	NaN	NaN	NaN	monographic	http
3	472	NaN	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	Appleyard, Ernest Silvanus.	NaN	NaN	NaN	NaN	monographic	http

`df.head()` : berfungsi untuk menampilkan baris awal dari dataFrame, secara default bila tdk diberi parameter akan menampilkan 5 baris data

Hands-on Data Cleaning: Membuang (drop) Kolom

- Melihat pada lima entri pertama dengan perintah `head()`.
- Dapat dilihat bahwa beberapa kolom memberikan informasi tambahan yang akan membantu perpustakaan tetapi tidak terlalu deskriptif tentang buku itu sendiri: `Edition Statement`, `Corporate Author`, `Corporate Contributors`, `Former owner`, `Engraver`, `Issuance type` and `Shelfmarks`.
- Kita drop kolom-kolom tsb dengan perintah:

```
to_drop = ['Edition Statement',
           'Corporate Author',
           'Corporate Contributors',
           'Former owner',
           'Engraver',
           'Contributors',
           'Issuance type',
           'Shelfmarks']
```

```
df.drop(to_drop, inplace=True, axis=1)
```

Kita de;nisikan daftar (list) nama dari semua kolom yang ingin kita drop. Kemudian jalankan perintah fungsi `drop()`.

dengan parameter `inplace` bernilai `True` dan parameter `axis` bernilai `1`, di mana 1 adalah angka sumbu (0 untuk baris dan 1 untuk kolom.)

Hands-on Data Cleaning: Membuang (drop) Kolom

- Inspeksi ulang `DataFrame`, kolom yang tidak diinginkan sudah dibuang

```
df.head()
```

	Identifier	Place of Publication	Date of Publication	Publisher	Title	Author	Flickr URL
0	206	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	http://www.flickr.com/photos/britishlibrary/ta...
1	216	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
2	218	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of "All for Gr...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
3	472	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...
4	480	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...

Hands-on Data Cleaning: Membuang (drop) Kolom

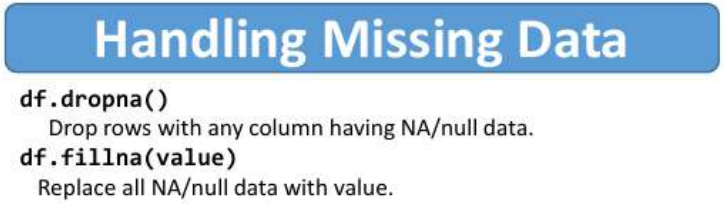
- Alternatif utk membuang kolom, dengan meneruskannya langsung ke parameter columns daripada memisahkan label-label yang mau dibuang:

```
df.drop(colums=to_drop, inplace=True)
```

inplace=True adalah perintah menimpa kolom dan menyimpan kolom/fitur yang dimanipulasi (dlm hal ini di drop)

- Sintak ini lebih intuitif dan mudah dibaca dibanding sintak sebelumnya?

```
df.drop(to_drop, inplace=True, axis=1)
```



Hands-on Data Cleaning: Mengubah Indeks di DataFrame

- Index dalam Pandas memperluas fungsionalitas array NumPy untuk memungkinkan pemotongan (slicing) dan pelabelan yang lebih fleksibel. Dalam banyak kasus, akan sangat membantu jika menggunakan ;eld pengenal data yang bernilai unik sebagai indeksnya.
- Sebagai contoh, dengan dataset di slide sebelumnya, praktiknya saat pustakawan mencari record, biasanya akan memasukan identi;er unik suatu buku:

```
df['Identifier'].is_unique
```

```
True
```

`df['Identi er']`: slicing/seleksi kolom/;eld yang akan di eksekusi.

`.is_unique`: function untuk mengecek nilai unik

Hands-on Data Cleaning: Mengubah Indeks di DataFrame

- Gantikan indeks yang ada pada kolom ini menggunakan `set_index` :

`.set_index()`: function untuk merubah index dengan diikuti parameter kolom yang akan dipilih untuk dijadikan index

```
df = df.set_index('Identifier')
df.head()
```

Identifier	Place of Publication	Date of Publication	Publisher	Title	Author	Flickr URL
206	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	http://www.flickr.com/photos/britishlibrary/ta...
216	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
218	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of "All for Gr...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
472	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...
480	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...

Technical Detail: Unlike primary keys in SQL, a Pandas `Index` doesn't make any guarantee of being unique, although many indexing and merging operations will notice a speedup in runtime if it is.

Hands-on Data Cleaning: Mengubah Indeks di DataFrame

- Kami dapat mengakses setiap records dengan cara yang mudah dengan `loc[]`. Cara ini digunakan untuk *label-based indexing*, yaitu memberi label suatu baris atau kolom tanpa memperhatikan posisi/lokasinya.

```
df.loc[206]
```

```
Place of Publication                                        London
Date of Publication                                     1879 [1878]
Publisher                                          S. Tinsley & Co.
Title                                  Walter Forbes. [A novel.] By A. A
Author                                                        A. A.
Flickr URL              http://www.flickr.com/photos/britishlibrary/ta...
Name: 206, dtype: object
```

- Dengan kata lain, 206 adalah label pertama dari indeks. Utk mengakses berdasarkan posisinya, gunakan `df.iloc[]`

loc: untuk seleksi dengan menggunakan label/bilangan bulat
iloc: untuk seleksi dengan menggunakan bilangan bulat
contoh penggunaan lain:
iin.iloc[::3,:].head(10): untuk Memilih baris kelipatan 3, dengan semua kolom dan menampilkan 10 data pertama.

Hands-on Data Cleaning: Mengubah Indeks di DataFrame

- Pada slide sebelumnya, Indeks yang digunakan adalah `RangeIndex`: integer mulai dari 0, analog dengan range di Python. Dengan meneruskan nama kolom ke `set_index`, maka indeks telah diubah ke nilai dalam Identi;er.

- Diperhatikan pada langkah sebelumnya bahwa telah dilakukan penetapan kembali variabel ke objek yang dikembalikan oleh metode dengan df = `df.set_index(...)`. Ini karena, secara default, metode mengembalikan salinan objek yang dimodi;kasi dan tidak membuat perubahan secara langsung ke objek. Hal ini dapat dihindari dengan mengatur parameter `inplace`:

```
df.set_index('Identifier', inplace=True)
```

Hands-on Data Cleaning: Merapihkan *Fields* dalam Data

- Slide sebelumnya telah dibuang bbrp kolom tidak penting dan diubah indeks pada `DataFrame` hingga menjadi lebih masuk akal.
- Selanjutnya, akan dibersihkan kolom tertentu dan mengubah menjadi bentuk/format yang seragam hingga dataset lebih mudah dipahami dan memastikan konsistensi. Dalam slide berikutnya akan dibersihkan `Date of Publication` dan `Place of Publication`.
- Dalam inspeksi, semua tipe data saat ini adalah objek `dtype` yang analog dengan `str` di native Python

Hands-on Data Cleaning: Merapihkan *Fields* dalam Data

- Cara ini dilakukan sebagai rangkuman saat setiap ;eld tidak dapat dirapihkan sebagai data numerik atau data kategorik dan data yang digunakan cukup "kotor" atau "berantakan".

```
df.dtypes.value_counts()
```

```
object    6
dtype: int64
```

Hands-on Data Cleaning: Merapihkan *Fields* dalam Data

- Satu kolom yang masuk akal untuk menerapkan nilai numerik adalah tanggal publikasi sehingga kita dapat melakukan perhitungan di awal:

- Buku tertentu hanya memiliki satu tanggal publikasi. Oleh karena itu perlu dilakukan hal berikut:
  - Hilangkan tanggal lain dalam kurung siku, `1879[1878]`
  - Konversi rentang tanggal ke "`start date`", `1860-63; 1839, 38-54`
  - Hilangkan tanggal yang tidak jelas dan gantikan dengan `NaN` NumPy, [1879?] -> `NaN`
  - Konversi string nan ke nilai NaN NumPy

```
df.loc[1905:, 'Date of Publication'].head(10)
```

```
Identifier
1905           1888
1929    1839, 38-54
2836           1897
2854           1865
2956        1860-63
2957           1873
3017           1866
3131           1899
4598           1814
4884           1820
Name: Date of Publication, dtype: object
```

`df.loc[1905:, 'Nama Field']`: digunakan untuk mengakses index mulai dari index 1905 dengan output hanya pada ;eld tanggal publikasi

`.head(10)`: function untuk menampilkan baris awal dataFrame dengan parameter hingga index ke 10 atau 10 baris data

Hands-on Data Cleaning: Merapihkan *Fields* dalam Data

- Mensintesis pola-pola ini, manfaatkan ekspresi reguler (Regex) tunggal untuk mengekstrak tahun publikasi.

```
regex = r'^(\d{4})'
```

- perintah `\d` mewakili sebarang digit dan `{4}` mengulangi aturan (rule) sebanyak empat kali. Kararakter `^` sesuai dengan awal string, dan tanda dalam kurung `()` menunjukkan *capruring group* yang memberikan sinyal ke Pandas bahwa akan dilakukan ekstraksi bagian Regex tersebut.

Hands-on Data Cleaning: Regex di Pandas

regex (Regular Expressions) Examples	
`'\.'`	Matches strings containing a period '.'
`'Length$'`	Matches strings ending with word 'Length'
`'^Sepal'`	Matches strings beginning with the word 'Sepal'
`'^x[1-5]$'`	Matches strings beginning with 'x' and ending with 1,2,3,4,5
`'^(?!Species$).*'`	Matches strings except the string 'Species'

`df.loc[:,'x2':'x4']`

Select all columns between x2 and x4 (inclusive).

`df.iloc[:,[1,2,5]]`

Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[df['a'] > 10, ['a','c']]`

Select rows meeting logical condition, and only the specific columns .

Hands-on Data Cleaning: Merapihkan *Fields* dalam Data

- Coba jalankan regex di dataset

```
extr = df['Date of Publication'].str.extract(r'^(\d{4})', expand=False)
extr.head()
```

```
Identifier
206    1879
216    1868
218    1869
472    1851
480    1857
Name: Date of Publication, dtype: object
```

Further Reading: Not familiar with regex? You can inspect the expression above at regex101.com and learn all about regular expressions with Regular Expressions: Regexes in Python.

Mengekstrak data untuk setiap string subjek hasil tangkapan variabel regex dari kolom `Date of Publication`

`expand=False`: Jika Benar, kembalikan DataFrame dengan satu kolom per grup tangkapan. Jika Salah, kembalikan Seri/Indeks jika ada satu grup tangkapan atau DataFrame jika ada beberapa grup tangkapan.

Hands-on Data Cleaning: Merapihkan *Fields* dalam Data

- Secara teknis, kolom tsb masih memiliki dtype = object, namun dengan mudah kita dapatkan versi numeriknya dengan perintah pd.to_numeric

```
df['Date of Publication'] = pd.to_numeric(extr)
df['Date of Publication'].dtype
```

```
dtype('float64')
```

- Ini menghasilkan sekitar 1/10 nilai yang hilang, cost yang cukup kecil dampaknya untuk saat ini karena dapat melakukan perhitungan pada nilai valid yang tersisa:

```
df['Date of Publication'].isnull().sum() / len(df)
```

```
0.11717147339205986
```

Hands-on Data Cleaning: Membersihkan Kolom dengan Kombinasi metode `str` dengan NumPy

- Slide sebelumnya dibahas penggunaan `df['Date of Publication'].str`. Atribut ini adalah cara akses cepat operasi string di Pandas yang menyerupai operasi pada native Python atau mengkompilasi regex seperti `.split()`, `.replace()`, dan `.capitalize()`.
- Utk membersihkan ;eld `Place of Publication`, kombinasikan metode `str` di Panda dengan fungsi `np.where` di NumPy yang mirip dengan bentuk vektor dari makro `IF()` di Excell, dengan sintak berikut:

```
np.where(condition, then, else)
```

Hands-on Data Cleaning: Membersihkan Kolom dengan Kombinasi metode `str` dengan NumPy

```
np.where(condition, then, else)
```

- condition mirip dengan objek array atau Boolean .then adalah nilai yang digunakan jika condition mengevaluasi menjadi True, dan else untuk mengevaluasi nilai selainnya.
- .where membawa tiap elemen dalam objek digunakan untuk condition dan memeriksa elemen tertentu menjadi True dalam konteks kondisi dan mengembalikan ndarray terdiri dari then atau else, tergantung pada prakteknya.

Hands-on Data Cleaning: Membersihkan Kolom dengan Kombinasi metode `str` dengan NumPy

- Dapat juga dituliskan dalam bersarang (nested) menjadi pernyataan *If-Then*, memungkinkan menghitung nilai berbasiskan kondisi berganda:

```
Python                                                    >>>

>>> np.where(condition1, x1,
             np.where(condition2, x2,
                 np.where(condition3, x3, ...)))
```

- Kemudian, dapat digunakan dua fungsi tsb untuk membersihkan ;eld Place of Publication karena kolom tsb memiliki objek string. Berikut adalah isi dari kolom:

Hands-on Data Cleaning: Membersihkan Kolom dengan Kombinasi metode `str` dengan NumPy

```
df['Place of Publication'].head(10)
```

```
Identifier
206                                        London
216                        London; Virtue & Yorston
218                                        London
472                                        London
480                                        London
481                                        London
519                                        London
667     pp. 40. G. Bryan & Co: Oxford, 1898
874                                       London]
1143                                       London
Name: Place of Publication, dtype: object
```

- Dilihat pada hasil di atas, ;eld `place of publication` masih ada informasi yang tidak penting. Jika dilihat lebih teliti, kasus ini untuk beberapa baris yang `place of publication` -nya di "London" dan "Oxford"

Hands-on Data Cleaning: Membersihkan Kolom dengan Kombinasi metode `str` dengan NumPy

```
df.loc[4157862]
```

```
Place of Publication                          Newcastle-upon-Tyne
Date of Publication                                        1867.0
Publisher                                             T. Fordyce
Title                  Local Records; or, Historical Register of rema...
Author                     FORDYCE, T. - Printer, of Newcastle-upon-Tyne
Flickr URL             http://www.flickr.com/photos/britishlibrary/ta...
Name: 4157862, dtype: object
```

```
df.loc[4159587]
```

```
Place of Publication                          Newcastle upon Tyne
Date of Publication                                        1834.0
Publisher                                          Mackenzie & Dent
Title                  An historical, topographical and descriptive v...
Author                                          Mackenzie, E. (Eneas)
Flickr URL             http://www.flickr.com/photos/britishlibrary/ta...
Name: 4159587, dtype: object
```

- Pada dua entri di samping, dua buku diterbitkan di tempat yang sama (newcastle upon tyne) namun salah satunya memilik tanda hubung (-)
- Untuk membersihkan kolom ini dalam sekali jalan, gunakan `str.contains()` untuk mendapatkan Boolean mask.

```
pub = df['Place of Publication']
london = pub.str.contains('London')
london[:5]
```

```
Identifier
206     True
216     True
218     True
472     True
480     True
Name: Place of Publication, dtype: bool
```

```
oxford = pub.str.contains('Oxford')
```

Hands-on Data Cleaning: Membersihkan Kolom dengan Kombinasi metode `str` dengan NumPy

- Kombinasikan dengan `np.where`:

```
df['Place of Publication'] = np.where(london, 'London',
                                      np.where(oxford, 'Oxford',
                                               pub.str.replace('-', ' ')))
df['Place of Publication'].head()
```

```
Identifier
206     London
216     London
218     London
472     London
480     London
Name: Place of Publication, dtype: object
```

- Di sini, fungsi `np.where` berbentuk struktur nested, dimana condition berbentuk Series dari Boolean dengan `str.contains()`. Metode `contains()` bekerja mirip dengan keyword in yang digunakan untuk mencari kejadian suatu entitas dalam kondisi pengulangan iterasi (atau substring dalam suatu string)

Hands-on Data Cleaning: Membersihkan Kolom dengan Kombinasi metode `str` dengan NumPy

- Pergantian tanda hubung (hypen) dengan spasi dengan `str.replace()` dan re-assign ke kolom dalam `DataFrame`.
- Walau pada kenyataan masih banyak dataset ini (kolom dan baris) yang "kotor", namun dalam contoh di sini hanya dibahas pada dua kolom

Hands-on Data Cleaning: Membersihkan Kolom dengan Kombinasi metode `str` dengan NumPy

- Coba periksa kembali untuk lima entri pertama, hasilnya akan lebih rapih dan "bersih" dibandingkan dataset awal sebelum dilakukan *cleaning dara*.

```
df.head()
```

Identifier	Place of Publication	Date of Publication	Publisher	Title	Author	Flickr URL
206	London	1879.0	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	http://www.flickr.com/photos/britishlibrary/ta...
216	London	1868.0	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
218	London	1869.0	Bradbury, Evans & Co.	Love the Avenger. By the author of "All for Gr...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
472	London	1851.0	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...
480	London	1857.0	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...

Note: At this point, `Place of Publication` would be a good candidate for conversion to a `Categorical` dtype, because we can encode the fairly small unique set of cities with integers. (*The memory usage of a Categorical is proportional to the number of categories plus the length of the data; an object dtype is a constant times the length of the data.*)

Hands-on Data Cleaning: Membersihkan Seluruh Dataset dengan Fungsi `applymap`

- Pada situasi tertentu, data berantakan alias "kotor" tidak hanya berlaku di sebagian kolom atau baris (record) tapi menyebar ke banyak bagian dataset.
- Cara berikut dapat diterapkan untuk semua cell atau elemen di `DataFrame` (dataset).
- Metode `.applymap()` dapat diterapkan, dimana similar dengan fungsi built-in yaitu fungsi `map()`.

Hands-on Data Cleaning: Membersikan seluruh Dataset

- Terapkan fungsi `applymap()` pada ;le "`university_towns.txt`":

Shell

```
$ head Datasets/univerisity_towns.txt
Alabama[edit]
Auburn (Auburn University)[1]
Florence (University of North Alabama)
Jacksonville (Jacksonville State University)[2]
Livingston (University of West Alabama)[2]
Montevallo (University of Montevallo)[2]
Troy (Troy University)[2]
Tuscaloosa (University of Alabama, Stillman College, Shelton State)[3][4]
Tuskegee (Tuskegee University)[5]
Alaska[edit]
```

Perintah shell

- Dapat dilihat di atas, bahwa nama negara bagian(state) diikuti dengan kota asal universitas:
  StateA TownA1 TownA2 StateB TownB1 TownB2....
  dan memiliki substring "`[edit]`"

Hands-on Data Cleaning: Membersihkan seluruh Dataset

- Kita dapat memanfaatkan pola ini dengan membuat list of (state, city) tuples dan *wrapping* daftar (list) dalam DataFrame.

```
university_towns = []
with open("C:/Users/Bayu/Documents/DTS 2021/Datasets/university_towns.txt") as file:
    for line in file:
        if '[edit]' in line:
            # Remember this `state` until the next is found
            state = line
        else:
            # Otherwise, we have a city; keep `state` as last-seen
            university_towns.append((state, line))

university_towns[:5]
```

```
[('Alabama[edit]\n', 'Auburn (Auburn University)[1]\n'),
 ('Alabama[edit]\n', 'Florence (University of North Alabama)\n'),
 ('Alabama[edit]\n', 'Jacksonville (Jacksonville State University)[2]\n'),
 ('Alabama[edit]\n', 'Livingston (University of West Alabama)[2]\n'),
 ('Alabama[edit]\n', 'Montevallo (University of Montevallo)[2]\n')]
```

- Kita dapat membungkus (wrap) daftar ini dalam DataFrame dan mengatur kolom sebagai "State" and "RegionName".
- Pandas akan mengambil setiap elemen dalam daftar dan mengatur "State" ke nilai kiri dan "RegionName" ke nilai kanan.
- Hasilnya adalah DataFrame sbb:

```
towns_df = pd.DataFrame(university_towns,
                        columns=['State', 'RegionName'])

towns_df.head()
```

	State	RegionName
0	Alabama[edit]\n	Auburn (Auburn University)[1]\n
1	Alabama[edit]\n	Florence (University of North Alabama)\n
2	Alabama[edit]\n	Jacksonville (Jacksonville State University)[2]\n
3	Alabama[edit]\n	Livingston (University of West Alabama)[2]\n
4	Alabama[edit]\n	Montevallo (University of Montevallo)[2]\n

Hands-on Data Cleaning: Membersihkan seluruh Dataset

- Pandas, mempermudah dalam pembersihan string dengan hanya membutuhkan nama state dan nama town dan dapat membuang lainnya. Selain dapat kembali menggunakan metode `.str()` di Pandas, dapat juga menggunakan `applymap()` untuk memetakan setiap elemen di DataFrame
- Perhatikan kasus sederhana pada contoh DataFrame berikut:

	0	1
0	mock	Dataset
1	python	pandas
2	real	python
3	numpy	clean

- Pada contoh di atas, setiap sel ("Mock", "Dataset", "Python", "Real", dll) adalah elemen. Oleh karena itu perintah `applymap()` akan menerapkan fungsi ke setiap elemen secara independen. Mari kita de;nisikan fungsi tsb:

Hands-on Data Cleaning: Membersihkan seluruh Dataset

- Fungsinya dide;nisikan berikut:

```
def get_citystate(item):
    if ' (' in item:
        return item[:item.find(' (')]
    elif '[' in item:
        return item[:item.find('[')]
    else:
        return item
```

- `applymap()` di Pandas hanya butuh satu parameter , yaitu fungsi yang diterapkan ke setiap elemen:

```
towns_df =  towns_df.applymap(get_citystate)
```

- Pertama, de;nisikan fungsi Python yang mengambil setiap elemen dari `DataFrame` sebagai parameternya. Di dalam fungsi, pengecekan dilakukan utk menentukan apakah ada elemen atau tidak!

Hands-on Data Cleaning: Membersihkan data dengan `applymap()`

- Tergantung pada pengecekan, nilai dikembalikan berdasarkan fungsi.
- Lalu, fungsi applymap() dipanggil pada objek yg ada. Sehingga kita dapatkan DataFrame yang relatif lebih rapih

```
towns_df.head()
```

	State	RegionName
0	Alabama	Auburn
1	Alabama	Florence
2	Alabama	Jacksonville
3	Alabama	Livingston
4	Alabama	Montevallo

Technical Detail: While it is a convenient and versatile method, `.applymap` can have significant runtime for larger datasets, because it maps a Python callable to each individual element. In some cases, it can be more efficient to do *vectorized* operations that utilize Cython or NumPY (which, in turn, makes calls in C) under the hood.

Hands-on Data Cleaning: Mengganti Nama Kolom

- Seringkali dalam dataset yang dimiliki ada nama kolom yang suli utk dipahami atau informasi yang tidak penting dalam beberapa baris awal/akhir, misal de;nisi istilah atau footnotes.
- Oleh karena itu dapat dilakukan penggantian nama dan melewatkan beberapa baris sehingga bisa dilakukan analisis informasi dari baris yang benar atau dapat dipahami.
- Kita akan lakukan utk lima baris awal dataset "`olympic.csv`":

Shell

```
$ head -n 5 Datasets/olympics.csv
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
,? Summer,01 !,02 !,03 !,Total,? Winter,01 !,02 !,03 !,Total,? Games,01 !,02 !,03 !,C
Afghanistan (AFG),13,0,0,2,2,0,0,0,0,0,13,0,0,2,2
Algeria (ALG),12,5,2,8,15,3,0,0,0,0,15,5,2,8,15
Argentina (ARG),23,18,24,28,70,18,0,0,0,0,41,18,24,28,70
```

Hands-on Data Cleaning: Mengganti Nama Kolom

- Kemudian, baca dalam DataFrame di Pandas:

```
olympics_df = pd.read_csv("C:/Users/Bayu/Documents/DTS 2021/Datasets/olympics.csv")
olympics_df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NaN	? Summer	01 !	02 !	03 !	Total	? Winter	01 !	02 !	03 !	Total	? Games	01 !	02 !	03 !	Combined total
1	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2
2	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15
3	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70
4	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12

- Hasilnya berantakan! Kolom adalah bentuk string integer indeks 0. Baris yang harusnya sebegai header pada `olympics_df.iloc[0]`. Hal ini terjadi karena ;le CSV mulai dengan 0, 1, 2, ..., 15.
- Dan, jika kita ke sumber dataset ini, akan terlihat NaN yang ada harusnya berisikan "Country" dan "Summer" maksudnya adala"Summer Games" dan "01!" harusnya adalah "Gold" , dll.

Hands-on Data Cleaning: Mengganti Nama Kolom

- Oleh karena itu, hal berikut yang perlu dilakukan:
  - Melewatkan (skip) satu baris dan atur header sebagai baris pertama (indeks-0)
  - Mengganti Nama Kolom
- Melewatkan baris dan atur header dapat dilakukan pada saat membaca ;le CSV dengan mempassing beberapa parameter ke fungsi `read_csv()`.
- Fungsi `read_csv()` memerlukan banyak parameter opsional, namun utk kasus ini hanya diperlukan satu (header) yang dihilangkan pada baris ke-0, dengan hasil sbb:

Hands-on Data Cleaning: Mengganti Nama Kolom

- Hasil fungsi `read_csv()` dan menghilangkan satu baris (header)

```
olympics_df = pd.read_csv("C:/Users/Bayu/Documents/DTS 2021/Datasets/olympics.csv", header=1)
olympics_df.head()
```

	Unnamed: 0	? Summer	01 !	02 !	03 !	Total	? Winter	01 !.1	02 !.1	03 !.1	Total.1	? Games	01 !.2	02 !.2	03 !.2	Combined total
0	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2
1	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15
2	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70
3	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12
4	Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	3	4	5	12

- Sekarang, yang tampak di samping adalah sekumpulan baris yang benar sebagai header dan semua baris yang tidak dibutuhkan telah dihilangkan.
- Pandas telah merubah nama kolom yang mengandung nama "countries" dari NaN menjadi Unnamed:0

Hands-on Data Cleaning: Mengganti Nama Kolom

- Utk mengganti nama kolom, digunakan metode `rename()` DataFrame yg memungkinkan memberi label pada axis berdasarkan pemetaan (dalam kasus ini yaitu dict)
- Mulai dengan mende;nisikan suatu kamus yang memetakan nama kolom saat ini sebagi kunci ke yang lebih dapat digunakan”

```
new_names =  {'Unnamed: 0': 'Country',
              '? Summer': 'Summer Olympics',
              '01 !': 'Gold',
              '02 !': 'Silver',
              '03 !': 'Bronze',
              '? Winter': 'Winter Olympics',
              '01 !.1': 'Gold.1',
              '02 !.1': 'Silver.1',
              '03 !.1': 'Bronze.1',
              '? Games': '# Games',
              '01 !.2': 'Gold.2',
              '02 !.2': 'Silver.2',
              '03 !.2': 'Bronze.2'}
```

Hands-on Data Cleaning: Mengganti Nama Kolom

- Kemudian, panggil fungsi `rename()` pada objek dimaksud:

```
olympics_df.rename(columns=new_names, inplace=True)
```

- Atur `inplace` menjadi `True`, dengan hasil sbb:

```
olympics_df.head()
```

	Country	Summer Olympics	Gold	Silver	Bronze	Total	Winter Olympics	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2	Bronze.2	Combined total
0	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2
1	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15
2	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70
3	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12
4	Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	3	4	5	12

Referensi

- Krensky P. Data Pre Tools: Goals, Bene;ts, and The Advantage of Hadoop. Aberdeen Group Report. July 2015
- SAS. Data Preparation Challenges Facing Every Enterprise. ebook. December 2017
- https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=6e9aa0e36f63
- https://improvado.io/blog/what-is-data-preparation
- https://searchenterpriseai.techtarget.com/feature/Data-preparation-for-machine-learning-still-requires-humans?
- https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e
- CRISP-DM

Tools Lab Online

- jupyter notebook
- scikit-learn
- pandas
- numpy

Ringkasan

- Data preparation memiliki sebutan lain, diantaranya data pre-processing, data cleaning, data manipulation,
- Data preparation mengambil porsi kerja terbanyak dalam data science 60-80%
- Data preparation membutuhkan ketelitian dan kesabaran/kerajinan dari peneliti DS, terutama pemula
- Data Validation merupakan tahapan kritikal dari DS namun sering diabaikan para peneliti
- Seleksi Fitur harus dilakukan di awal tahapan data preparation setelah melakukan penentuan metode/teknik sampling
- Data cleaning merupakan pekerjaan yang sangat memerlukan keahlian teknik DS terkait penggunakan tools dan coding
- Kebersihan data merupakan sarat mutlak utk Model Prediksi yang Baik.

Terima Kasih