AI VIET NAM
@aivietnam.edu.vn

Toward Data Science
Pandas 2

Year 2023

Create an example DataFrame

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Create an example DataFrame

```
[74] df = pd.DataFrame({'col one':[100, 200], 'col two':[300, 400]})
     df
```

	col one	col two
0	100	300
1	200	400

Define the dataframe yourself

```
pd.DataFrame(np.random.rand(4, 8), columns=list('abcdefgh'))
```

	a	b	c	d	e	f	g	h
0	0.211025	0.148126	0.679437	0.287645	0.035275	0.285124	0.313314	0.121054
1	0.361451	0.996439	0.401521	0.234080	0.103908	0.838070	0.256617	0.584056
2	0.243553	0.059943	0.055316	0.450530	0.190026	0.782297	0.856126	0.569080
3	0.600643	0.975051	0.519675	0.140729	0.668187	0.905942	0.035337	0.088676

Define the columns name with random value

```
[76] pd.util.testing.makeDataFrame().head()
```

	A	B	C	D
xIH13iGHP4	0.919202	-1.084965	-0.330164	-1.681841
aoHAyo00PI	1.544368	0.193910	-0.398716	-0.267523
zmuQV32no7	0.345481	1.011625	0.335001	1.620216
GC0TzYhnKm	0.841254	-1.780052	-0.098154	0.430689
D4PEbuD1mA	0.623624	-0.089280	-2.289458	0.765787

Build-in random dataframe

AI VIET NAM
@aivietnam.edu.vn

Create an example DataFrame

```
[78] pd.util.testing.makeTimeDataFrame().head()
```

	A	B	C	D
2000-01-03	-0.133812	1.550887	-0.026166	0.686443
2000-01-04	1.036991	1.307356	-1.483159	-1.304558
2000-01-05	0.766995	-0.075220	0.408392	-0.267696
2000-01-06	-1.199219	-1.640139	0.487682	-0.253413
2000-01-07	0.681742	-0.897427	-0.089646	0.004583

Build-in random dataframe with date-time index

```
pd.util.testing.makeMixedDataFrame()
```

	A	B	C	D
0	0.0	0.0	foo1	2009-01-01
1	1.0	1.0	foo2	2009-01-02
2	2.0	0.0	foo3	2009-01-05
3	3.0	1.0	foo4	2009-01-06
4	4.0	0.0	foo5	2009-01-07

Build-in random dataframe with random properties

```
[80] # And many more
     [x for x in dir(pd.util.testing) if x.startswith('make')]
```

```
['makeBoolIndex',
 'makeCategoricalIndex',
 'makeCustomDataframe',
 'makeCustomIndex',
 'makeDataFrame',
```

You can check all the sample dataframe here

Reverse Row/Column order

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Reverse Row/Column order

Reverse Row/Column order

Row

[: : -1] mean iterate in reverse

```
[81] data.head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

```
data.loc[::-1].head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True

AI VIET NAM
@aivietnam.edu.vn

Reverse Row/Column order

```
[82] data.loc[::-1].head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True

```
data.loc[::-1].reset_index(drop=True).head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True
1	720	HoopaHoopa Unbound	Psychic	Dark	680								True
2	720	HoopaHoopa Confined	Psychic	Ghost	600								True
3	719	DiancieMega Diancie	Rock	Fairy	700								True
4	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True

use reset_index(drop=True) to reset index order

AI VIET NAM
@aivietnam.edu.vn

Reverse Row/Column order

Column

```
[84] data.loc[:, ::-1].head()
```

	Legendary	Generation	Speed	Sp. Def	Sp. Atk	Defense	Attack	HP	Total	Type 2	Type 1	Name	#
0	False	1	45	65	65	49	49	45	318	Poison	Grass	Bulbasaur	1
1	False	1	60	80	80	63	62	60	405	Poison	Grass	Ivysaur	2
2	False	1	80	1(								Venusaur	3
3	False	1	80	120	122	123	100	80	625	Poison	Grass	VenusaurMega Venusaur	3
4	False	1	65	50	60	43	52	39	309	NaN	Fire	Charmander	4

With columns [:, : : -1]

Select column by data type

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Select column by data type

Select column by data type

```
data.dtypes
```

```
#               int64
Name           object
Type 1         object
Type 2         object
Total           int64
HP              int64
Attack          int64
Defense         int64
Sp. Atk         int64
Sp. Def         int64
Speed           int64
Generation      int64
Legendary        bool
dtype: object
```

Select all columns that are int or float

```
[86] data.select_dtypes(include='number').head()
```

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
0	1	318	45	49	49	65	65	45	1
1	2	405	60	62	63	80	80	60	1
2	3	525	80	82	83	100	100	80	1
3	3	625	80	100	123	122	120	80	1
4	4	309	39	52	43	60	50	65	1

AI VIET NAM
@aivietnam.edu.vn

Select column by data type

```
[88] data.select_dtypes(include=['number', 'object']).head()
```

We can select multiple type by passing a list

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1

```
data.select_dtypes(exclude=['number', 'object']).head()
```

	Legendary
0	False
1	False
2	False
3	False
4	False

Or we can choose to exclude some type

Convert strings to numbers

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Convert strings to numbers

Convert strings to numbers

```
df = pd.DataFrame({'col_one':['1.1', '2.2', '3.3'],
                   'col_two':['4.4', '5.5', '6.6'],
                   'col_three':['7.7', '8.8', '-']})
df
```

	col_one	col_two	col_three
0	1.1	4.4	7.7
1	2.2	5.5	8.8
2	3.3	6.6	-

This data is of type ‘str’

```
[91] df.dtypes
```

```
col_one      object
col_two      object
col_three    object
dtype: object
```

```
[92] df.astype({'col_one':'float', 'col_two':'float'}).dtypes
```

```
col_one      float64
col_two      float64
col_three     object
dtype: object
```

We can change the first and second columns to float but the third give error

```
[93] df.astype({'col_one':'float', 'col_two':'float', 'col_three': 'float'}).dtypes
```

AI VIET NAM
@aivietnam.edu.vn

Convert strings to numbers

```
[94] pd.to_numeric(df.col_three, errors='coerce')

0    7.7
1    8.8
2    NaN
Name: col_three, dtype: float64
```

A better way is to use pd.to_numeric

The dash will be converted to NaN

```
[95] pd.to_numeric(df.col_three, errors='coerce').fillna(0)

0    7.7
1    8.8
2    0.0
Name: col_three, dtype: float64
```

```
df = df.apply(pd.to_numeric, errors='coerce').fillna(0)
df
```

We can choose to fill NaN with 0 if desired

	col_one	col_two	col_three
0	1.1	4.4	7.7
1	2.2	5.5	8.8
2	3.3	6.6	0.0

```
[97] df.dtypes

col_one      float64
col_two      float64
col_three    float64
dtype: object
```

Dealing with large Dataset

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Dealing with large Dataset

Dealing with large Dataset

Check memery usage

```
data.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   #           800 non-null    int64
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      414 non-null    object
 4   Total       800 non-null    int64
 5   HP          800 non-null    int64
 6   Attack      800 non-null    int64
 7   Defense     800 non-null    int64
 8   Sp. Atk     800 non-null    int64
 9   Sp. Def     800 non-null    int64
 10  Speed       800 non-null    int64
 11  Generation  800 non-null    int64
 12  Legendary   800 non-null    bool
dtypes: bool(1), int64(9), object(3)
memory usage: 194.7 KB
```

Some dataframe can be extremely large

Current memory usage: 194.7 KB

AI VIET NAM
@aivietnam.edu.vn

Dealing with large Dataset

Load in specific Columns

```
[99] cols = ['Name', 'Type 1']
     small_data = pd.read_csv('/content/Pokemon.csv', usecols=cols)
     small_data.head()
```

	Name	Type 1
0	Bulbasaur	Grass
1	Ivysaur	Grass
2	Venusaur	Grass
3	VenusaurMega Venusaur	Grass
4	Charmander	Fire

We can choose to only load the columns we need

```
small_data.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    800 non-null    object
 1   Type 1  800 non-null    object
dtypes: object(2)
memory usage: 100.3 KB
```

memory usage reduce to: 100.3 KB

AI VIET NAM
@aivietnam.edu.vn

Dealing with large Dataset

```
[101] dtypes = {'Type 1':'category'}
      smaller_data = pd.read_csv('/content/Pokemon.csv', usecols=cols, dtype=dtypes)
      smaller_data.head()
```

	Name	Type 1
0	Bulbasaur	Grass
1	Ivysaur	Grass
2	Venusaur	Grass
3	VenusaurMega Venusaur	Grass
4	Charmander	Fire

Some columns can be load as ‘category’ to reduce memory usage

```
smaller_data.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    800 non-null    object
 1   Type 1  800 non-null    category
dtypes: category(1), object(1)
memory usage: 54.1 KB
```

memory usage reduce to: 54.1 KB

AI VIET NAM
@aivietnam.edu.vn

Dealing with large Dataset

Load in Chunks

```
[103] chunks_df = pd.read_csv('/content/Pokemon.csv', chunksize=100)
```

We can choose to load in one chunk at a time

```
next(chunks_df)
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	88	Grimer	Poison	NaN	325	80	80	50	40	50	25	1	False
96	89	Muk	Poison	NaN	500	105	105	75	65	100	50	1	False
97	90	Shellder	Water	NaN	305	30	65	100	45	25	40	1	False
98	91	Cloyster	Water	Ice	525	50	95	180	85	45	70	1	False
99	92	Gastly	Ghost	Poison	310	30	35	30	100	35	80	1	False

100 rows × 13 columns

AI VIET NAM
@aivietnam.edu.vn

Dealing with large Dataset

```
[105] for i, df in enumerate(pd.read_csv('/content/Pokemon.csv', chunksize=100)):
          df.to_csv(f'/content/Pokemon_{i}.csv', index=False)
```

```
[106] df0 = pd.read_csv('/content/Pokemon_0.csv')
      df0.head()
```

We can save each chunk to seperate data files for later usage

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

```
[107] df1 = pd.read_csv('/content/Pokemon_1.csv')
      df1.head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	93	Haunter	Ghost	Poison	405	45	50	45	115				
1	94	Gengar	Ghost	Poison	500	60	65	60	130				
2	94	GengarMega Gengar	Ghost	Poison	600	60	65	80	170	95	130	1	False
3	95	Onix	Rock	Ground	385	35	45	160	30	45	70	1	False
4	96	Drowzee	Psychic	NaN	328	60	48	45	43	90	42	1	False

But how can we combine the data back?

Build a DataFrame from multiple files

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Build a DataFrame from multiple files

Build a DataFrame from multiple files (row-wise)

```
[108] df0.head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN					60	50	65	1	False

Collapse 5 child cells under Build a DataFrame from multiple files (row-wise) (Press <Shift> to also collapse sibling sections)

```
[109] df1.head()
```

Suppose we want to join these two dataframe row-wise

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	93	Haunter	Ghost	Poison	405	45	50	45	115	55	95	1	False
1	94	Gengar	Ghost	Poison	500	60	65	60	130	75	110	1	False
2	94	GengarMega Gengar	Ghost	Poison	600	60	65	80	170	95	130	1	False
3	95	Onix	Rock	Ground	385	35	45	160	30	45	70	1	False
4	96	Drowzee	Psychic	NaN	328	60	48	45	43	90	42	1	False

AI VIET NAM
@aivietnam.edu.vn

Build a DataFrame from multiple files

```
[151] df = pd.concat([df0, df1])
      df.head(5)
```

use pd.concat and pass in the list of dataframe we want to join

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

```
df[95:105]
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
95	88	Grimer	Poison	NaN	325	80	80	50	40	50	25	1	False
96	89	Muk	Poison	NaN	500	105	105	75	65	100	50	1	False
97	90	Shellder	Water	NaN	305	30	65	100	45	25	40	1	False
98	91	Cloyster	Water	Ice	525	50	95	180					
99	92	Gastly	Ghost	Poison	310	30	35	30	1				
0	93	Haunter	Ghost	Poison	405	45	50	45	1				
1	94	Gengar	Ghost	Poison	500	60	65	60	130	75	110	1	False
2	94	GengarMega Gengar	Ghost	Poison	600	60	65	80	170	95	130	1	False
3	95	Onix	Rock	Ground	385	35	45	160	30	45	70	1	False
4	96	Drowzee	Psychic	NaN	328	60	48	45	43	90	42	1	False

There is one issue, notice that the index got reset where the dataframes is joined

AI VIET NAM
@aivietnam.edu.vn

Build a DataFrame from multiple files

```
df = pd.concat([df0, df1], ignore_index=True)
df[95:105]
```

using ignore_index=True will resolve this issue

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
95	88	Grimer	Poison	NaN	325	80	80	50	40	50	25	1	False
96	89	Muk	Poison	NaN	500	105	105	75	65	100	50	1	False
97	90	Shellder	Water	NaN	305	30	65	100	45	25	40	1	False
98	91	Cloyster	Water	Ice	525	50	95	180	85	45	70	1	False
99	92	Gastly	Ghost	Poison	310	30	35	30	100	35	80	1	False
100	93	Haunter	Ghost	Poison	405	45	50	45	115	55	95	1	False
101	94	Gengar	Ghost	Poison	500	60	65	60	130	75	110	1	False
102	94	GengarMega Gengar	Ghost	Poison	600	60	65	80	170	95	130	1	False
103	95	Onix	Rock	Ground	385	35	45	160	30	45	70	1	False
104	96	Drowzee	Psychic	NaN	328	60	48	45	43	90	42	1	False

AI VIET NAM
@aivietnam.edu.vn

Build a DataFrame from multiple files

```
[115] data.to_csv('/content/Pokemon_first.csv', columns=data.columns[:6], index=False)
      data.to_csv('/content/Pokemon_second.csv', columns=data.columns[6:], index=False)
```

```
[116] df_f = pd.read_csv('/content/Pokemon_first.csv')
      df_f.head()
```

	#	Name	Type 1	Type 2	Total	HP
0	1	Bulbasaur	Grass	Poison	318	45
1	2	Ivysaur	Grass	Poison	405	60
2	3	Venusaur	Grass	Poison	525	80
3	3	VenusaurMega Venusaur	Grass	Poison	625	80
4	4	Charmander	Fire	NaN	309	39

For example we want to join these two dataframes columns-wise

```
df_s = pd.read_csv('/content/Pokemon_second.csv')
df_s.head()
```

	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	49	49	65	65	45	1	False
1	62	63	80	80	60	1	False
2	82	83	100	100	80	1	False
3	100	123	122	120	80	1	False
4	52	43	60	50	65	1	False

AI VIET NAM
@aivietnam.edu.vn

Build a DataFrame from multiple files

```
[118] df = pd.concat([df_f, df_s], axis='columns')
      df.head()
```

For this we need to include axis=‘colums’ param

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

Split a DataFrame into two random subsets

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Random Split DataFrame

Split a DataFrame into two random subsets

We sample 75% from our dataframe to be data_1

```
[119] len(data)
```

```
800
```

```
data_1 = data.sample(frac=0.75, random_state=1234)
np.sort(data_1.index)
```

```
array([  0,   2,   5,   6,   7,   8,   9,  11,  13,  16,  17,  19,  20,
        21,  22,  23,  24,  25,  27,  28,  29,  31,  33,  34,  35,  36,
        37,  38,  39,  40,  42,  43,  44,  47,  49,  50,  51,  52,  54,
        55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,  66,  67,
        69,  70,  71,  72,  73,  74,  77,  78,  79,  80,  83,  85,  88,
        89,  90,  91,  92,  93,  94,  95,  96,  97,  99, 100, 101, 102,
       104, 105, 106, 108, 109, 110, 111, 112, 113, 114, 115, 118, 121,
       122, 123, 124, 125, 126, 127, 128, 129, 131, 132, 133, 134, 137,
       138, 140, 141, 143, 144, 145, 146, 147, 148, 149, 151, 152, 153,
       155, 156, 157, 159, 160, 161, 162, 163, 166, 167, 168, 169, 170,
       171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 184,
       185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 198, 199, 200,
       201, 202, 203, 205, 206, 207, 213, 214, 215, 216, 217, 218, 219,
       220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232,
       234, 237, 238, 239, 240, 241, 242, 245, 247, 248, 249, 251, 252,
       253, 254, 255, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269,
       270, 271, 272, 273, 274, 276, 277, 278, 280, 281, 282, 285, 287,
       288, 289, 290, 296, 297, 298, 299, 300, 301, 302, 304, 305, 306,
       307, 308, 309, 310, 315, 319, 320, 321, 322, 323, 324, 326, 327,
       328, 329, 330, 331, 333, 335, 337, 338, 339, 341, 342, 344, 347,
       348, 349, 351, 352, 353, 355, 356, 357, 358, 359, 360, 361, 362,
       363, 364, 366, 367, 368, 369, 370, 373, 375, 376, 378, 380, 381,
       382, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395,
       397, 398, 399, 401, 402, 403, 405, 406, 407, 409, 410, 411, 412,
```

AI VIET NAM
@aivietnam.edu.vn

Random Split DataFrame

```
data_2 = data.drop(data_1.index)
np.sort(data_2.index)
```

```
array([  1,   3,   4,  10,  12,  14,  15,  18,  26,  30,  32,  41,  45,
        46,  48,  53,  68,  75,  76,  81,  82,  84,  86,  87,  98, 103,
       107, 116, 117, 119, 120, 130, 135, 136, 139, 142, 150, 154, 158,
       164, 165, 183, 195, 196, 197, 204, 208, 209, 210, 211, 212, 233,
       235, 236, 243, 244, 246, 250, 256, 257, 258, 259, 275, 279, 283,
       284, 286, 291, 292, 293, 294, 295, 303, 311, 312, 313, 314, 316,
       317, 318, 325, 332, 334, 336, 340, 343, 345, 346, 350, 354, 365,
       371, 372, 374, 377, 379, 383, 396, 400, 404, 408, 417, 425, 429,
       431, 434, 436, 439, 440, 442, 447, 448, 455, 459, 460, 465, 469,
       471, 476, 482, 487, 490, 493, 497, 506, 509, 510, 514, 520, 527,
       528, 531, 540, 546, 547, 550, 551, 553, 559, 561, 562, 568, 573,
       574, 577, 580, 591, 593, 595, 596, 597, 600, 601, 608, 610, 617,
       619, 625, 626, 633, 638, 640, 654, 655, 661, 663, 664, 671, 683,
       685, 686, 687, 689, 695, 702, 706, 707, 717, 723, 730, 731, 736,
       739, 740, 743, 745, 748, 751, 752, 756, 757, 763, 765, 768, 775,
       779, 784, 790, 792, 795])
```

Then we drop all data_1.index to be data_2

```
len(data_1) + len(data_2)
```

```
800
```

If we check, total len of two data is the same as the original data

Handle missing values

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Handle missing values

Handle missing values

```
data.isna().sum()
```

```
#               0
Name            0
Type 1          0
Type 2        386
Total           0
HP              0
Attack          0
Defense         0
Sp. Atk         0
Sp. Def         0
Speed           0
Generation      0
Legendary       0
dtype: int64
```

use isna().sum() to check total missing value in all columns

```
[124] data.isna().mean()
```

```
#             0.0000
Name          0.0000
Type 1        0.0000
Type 2        0.4825
Total         0.0000
HP            0.0000
Attack        0.0000
Defense       0.0000
Sp. Atk       0.0000
Sp. Def       0.0000
Speed         0.0000
Generation    0.0000
Legendary     0.0000
dtype: float64
```

use isna().mean() to check the proportion of missing value in all columns

AI VIET NAM
@aivietnam.edu.vn

Handle missing values

```
[158] data.dropna(axis='rows').head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	weak	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	normal	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80							
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	normal	123	122	120	80	1	False
6	6	Charizard	Fire	Flying	534	78	normal	78	109	85	100	1	False

We can choose to delete all rows/columns that have missing values

```
data.dropna(axis='columns', thresh=len(data)*0.6).head()
```

	#	Name	Type 1	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	525	80	82	83	100	100			
3	3	VenusaurMega Venusaur	Grass	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	309	39	52	43	60	50	65	1	False

We can choose the threshold too

AI VIET NAM
@aivietnam.edu.vn

Handle missing values

```
data.fillna('None Type').head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	None Type	309	39	52	43	60	50	65	1	False

We can choose to fill the missing value

```
data.interpolate().head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

Or interpolate the value by surrounding values

Split a string into multiple columns

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Split a string into multiple columns

Split a string into multiple columns

```
df = pd.DataFrame({'name':['John Arthur Doe', 'Jane Ann Smith'],
                   'location':['Los Angeles, CA', 'Washington, DC']})
df
```

	name	location
0	John Arthur Doe	Los Angeles, CA
1	Jane Ann Smith	Washington, DC

This is extremely useful when working with Name, Address, Date

```
[62] df[['first', 'middle', 'last']] = df.name.str.split(' ', expand=True)
df
```

	name	location	first	middle	last
0	John Arthur Doe	Los Angeles, CA	John	Arthur	Doe
1	Jane Ann Smith	Washington, DC	Jane	Ann	Smith

Reshape a MultiIndexed Series

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Reshape a MultiIndexed Series

```
data.groupby(['Type 1', 'Legendary'])['#'].count()
```

```
Type 1    Legendary
Bug       False        69
Dark      False        29
          True          2
Dragon    False        20
          True         12
Electric  False        40
          True          4
Fairy     False        16
          True          1
Fighting  False        27
Fire      False        47
          True          5
Flying    False         2
          True          2
Ghost     False        30
          True          2
Grass     False        67
          True          3
Ground    False        28
          True          4
Ice       False        22
          True          2
Normal    False        96
          True          2
Poison    False        28
Psychic   False        43
          True         14
Rock      False        40
          True          4
Steel     False        23
          True          4
Water     False       108
          True          4
Name: #, dtype: int64
```

What if we want True/False to be it own colums

AI VIET NAM
@aivietnam.edu.vn

Reshape a MultiIndexed Series

```
f = data.groupby(['Type 1', 'Legendary'])['#'].count().unstack()
f
```

Legendary	False	True
Type 1		
Bug	69.0	NaN
Dark	29.0	2.0
Dragon	20.0	12.0
Electric	40.0	4.0
Fairy	16.0	1.0
Fighting	27.0	NaN
Fire	47.0	5.0
Flying	2.0	2.0
Ghost	30.0	2.0
Grass	67.0	3.0
Ground	28.0	4.0
Ice	22.0	2.0
Normal	96.0	2.0
Poison	28.0	NaN
Psychic	43.0	14.0
Rock	40.0	4.0
Steel	23.0	4.0

Use .unstack() after .groupby()

Create a pivot table

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Create a pivot table

```
data.pivot_table(index='Type 1', columns='Legendary', values='Attack', aggfunc='mean')
```

Legendary	False	True
Type 1		
Bug	70.971014	NaN
Dark	86.862069	110.500000
Dragon	103.400000	126.666667
Electric	66.125000	98.750000
Fairy	57.187500	131.000000
Fighting	96.777778	NaN
Fire	82.191489	109.000000
Flying	50.000000	107.500000
Ghost	71.366667	110.000000
Grass	72.119403	97.666667
Ground	88.000000	150.000000
Ice	73.227273	67.500000
Normal	72.083333	140.000000
Poison	74.678571	NaN
Psychic	54.953488	122.142857
Rock	89.925000	122.250000
Steel	92.086957	96.250000
Water	72.777778	111.250000

We can achieve the same effect by using a pivot table

AI VIET NAM
@aivietnam.edu.vn

Create a pivot table

```
data.pivot_table(index='Type 1', columns='Legendary', values='Attack', aggfunc='mean', margins=True)
```

Legendary	False	True	All
Type 1			
Bug	70.971014	NaN	70.971014
Dark	86.862069	110.500000	88.387097
Dragon	103.400000	126.666667	112.125000
Electric	66.125000	98.750000	69.090909
Fairy	57.187500	131.000000	61.529412
Fighting	96.777778	NaN	96.777778
Fire	82.191489	109.000000	84.769231
Flying	50.000000	107.500000	78.750000
Ghost	71.366667	110.000000	73.781250
Grass	72.119403	97.666667	73.214286
Ground	88.000000	150.000000	95.750000
Ice	73.227273	67.500000	72.750000
Normal	72.083333	140.000000	73.469388
Poison	74.678571	NaN	74.678571
Psychic	54.953488	122.142857	71.456140
Rock	89.925000	122.250000	92.863636
Steel	92.086957	96.250000	92.703704
Water	72.777778	111.250000	74.151786
All	75.669388	116.676923	79.001250

The advantage of pivot table compare to using groupby and unstack is the ‘Margin’

Reshape a DataFrame from wide format to long format

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Reshape a MultiIndexed Series

Reshape a DataFrame from wide format to long format

```
f = data[['Name', 'HP', 'Attack']].iloc[:5]
f
```

	Name	HP	Attack
0	Bulbasaur	45	49
1	Ivysaur	60	62
2	Venusaur	80	82
3	VenusaurMega Venusaur	80	100
4	Charmander	39	52

What if we want ‘HP’ and ‘Attack’ to be the values of a columns name ‘Stats’?

```
[138] f2 = f.melt(id_vars='Name', var_name='Stats', value_name='Point')
      f2
```

	Name	Stats	Point
0	Bulbasaur	HP	45
1	Ivysaur	HP	60
2	Venusaur	HP	80
3	VenusaurMega Venusaur	HP	80
4	Charmander	HP	39
5	Bulbasaur	Attack	49
6	Ivysaur	Attack	62

Convert continuous data into categorical data

https://www.kaggle.com/datasets/abcsds/pokemon?resource=download

AI VIET NAM
@aivietnam.edu.vn

Continuous into Categorical

Convert continuous data into categorical data

data

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123					
4	4	Charmander	Fire	NaN	309	39	52	43					
...	...	...	...	...	...	...	...	...					
795	719	Diancie	Rock	Fairy	600	50	100	150					
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110					
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60					
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True

800 rows × 13 columns

What if we want 'Attack' to be categorized
< 50: 'weak'
50-100: 'normal'
100-150: 'strong'
>150: 'nani?!'

AI VIET NAM
@aivietnam.edu.vn

Continuous into Categorical

```
[141] df = data
      df['Attack'] = pd.cut(data['Attack'], bins=[0, 50, 100, 150, np.inf], labels=['weak', 'normal', 'strong', 'nani?!'])
```

```
df
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	weak	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	normal	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	normal	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	normal	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	normal	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	600	50	normal	150					
796	719	DiancieMega Diancie	Rock	Fairy	700	50	nani?!	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	strong	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	nani?!	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	600	80	strong	120	130	90	70	6	True

800 rows × 13 columns

Use pd.cut(<colums>, <bin>, <labels>)

Profile a DataFrame

AI VIET NAM
@aivietnam.edu.vn

Profile a DataFrame

Profile a DataFrame

```
data.head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	weak	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	normal	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	normal	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	normal	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	normal	43	60				

Want some quick EDA?

```
[146] data.describe()
```

pd.describe() too bad for you?

	#	Total	HP	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	800.000000	800.00000	800.000000	800.000000	800.000000	800.000000	800.000000	800.00000
mean	362.813750	435.10250	69.258750	73.842500	72.820000	71.902500	68.277500	3.32375
std	208.343798	119.96304	25.534669	31.183501	32.722294	27.828916	29.060474	1.66129
min	1.000000	180.00000	1.000000	5.000000	10.000000	20.000000	5.000000	1.00000
25%	184.750000	330.00000	50.000000	50.000000	49.750000	50.000000	45.000000	2.00000
50%	364.500000	450.00000	65.000000	70.000000	65.000000	70.000000	65.000000	3.00000
75%	539.250000	515.00000	80.000000	90.000000	95.000000	90.000000	90.000000	5.00000
max	721.000000	780.00000	255.000000	230.000000	194.000000	230.000000	180.000000	6.00000

AI VIET NAM
@aivietnam.edu.vn

Profile a DataFrame

```
!pip install pandas-profiling
from pandas_profiling import ProfileReport
```

```
[150] ProfileReport(data)
```

Summarize dataset: 100% 86/86 [00:14<00:00, 5.33it/s, Completed]

Generate report structure: 100% 1/1 [00:04<00:00, 4.55s/it]

Render HTML: 100% 1/1 [00:02<00:00, 2.27s/it]

Introducing 'pandas-profiling.ProfileReport'

AI VIET NAM
@aivietnam.edu.vn

Profile a DataFrame

Pandas Profiling Report

Overview | Variables | Interactions | Correlations | Missing values | Sample

Overview

Overview | Alerts 15 | Reproduction

Alerts

`Name` has a high cardinality: 800 distinct values	High cardinality
`#` is highly overall correlated with `Generation`	High correlation
`Total` is highly overall correlated with `HP` and 5 other fields	High correlation
`HP` is highly overall correlated with `Total`	High correlation
`Defense` is highly overall correlated with `Total` and 1 other fields	High correlation
`Sp. Atk` is highly overall correlated with `Total` and 2 other fields	High correlation
`Sp. Def` is highly overall correlated with `Total` and 2 other fields	High correlation
`Speed` is highly overall correlated with `Total`	High correlation
`Generation` is highly overall correlated with `#`	High correlation
`Legendary` is highly overall correlated with `Total` and 1 other fields	High correlation
`Legendary` is highly imbalanced (59.3%)	Imbalance
`Type 2` has 386 (48.2%) missing values	Missing

AI VIET NAM
@aivietnam.edu.vn

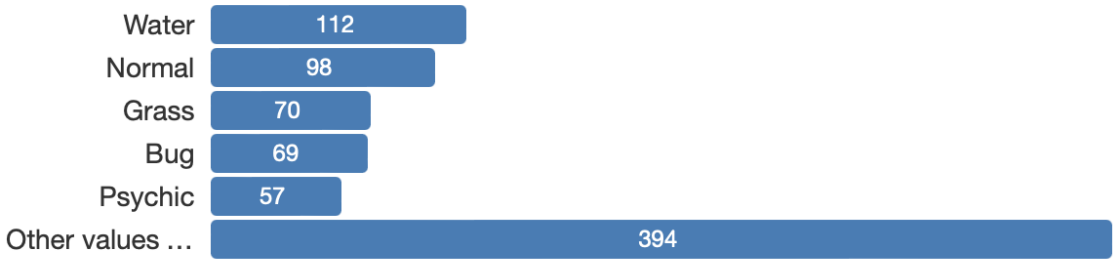
Profile a DataFrame

Pandas Profiling Report

Overview | Variables | Interactions | Correlations | Missing values | Sample

Type 1
Categorical

Distinct	18
Distinct (%)	2.2%
Missing	0
Missing (%)	0.0%
Memory size	6.4 KiB



More details

Overview | Categories | Words | Characters

Length

Max length	8
Median length	7
Mean length	5.26
Min length	3

Characters and Unicode

Total characters	4208
Distinct characters	28
Distinct categories	2
Distinct scripts	1
Distinct blocks	1

Unique

Unique	0
Unique (%)	0.0%

Sample

1st row	Grass
2nd row	Grass
3rd row	Grass
4th row	Grass
5th row	Fire

Profile a DataFrame