# **Reference guide: Data cleaning in Python**

This reference guide contains common functions and methods that data professionals use to clean data. The reference guide contains three different tables of useful tools, each grouped by cleaning category: missing data, outliers, and label encoding.

## Missing data

The following pandas functions and methods are helpful when dealing with missing data.

### [**df.info()**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.info.html#pandas.DataFrame.info)

* **Description:**  A DataFrame method that returns a concise summary of the dataframe, including a ‘non-null count,’ which helps you know the number of missing values
* **Example:**

import numpy as np

import pandas as pd

data = [['Mercury', 2440, 0], ['Venus', 6052, 0,], ['Earth', 6371, 1],

['Mars', 3390, 2], ['Jupiter', 69911, 80], ['Saturn', 58232, 83],

['Uranus', 25362, 27], ['Neptune', 24622, 14]

]

cols = ['Planet', 'radius\_km', 'moons']

df = pd.DataFrame(data, columns=cols)

print(df)

print()

df.info()

### [**df.isna()**](https://pandas.pydata.org/docs/reference/api/pandas.isna.html) **/ isnull()**

* **Description:** A pandas function that returns a same-sized Boolean array indicating whether each value is null (you can also use pd.isnull() as an alias). Note that this function also exists as a DataFrame method.
* **Example:**

**import numpy as np**

**import pandas as pd**

**data = [['Mercury', 2440, np.NaN], ['Venus', 6052, np.NaN,], ['Earth', 6371, 1],**

**['Mars', 3390, np.NaN], ['Jupiter', 69911, 80], ['Saturn', 58232, 83],**

**['Uranus', 25362, 27], ['Neptune', 24622, 14]**

**]**

**cols = ['Planet', 'radius\_km', 'moons']**

**df = pd.DataFrame(data, columns=cols)**

print(df)

print()

pd.isnull(df)

### [**pd.notna()**](https://pandas.pydata.org/docs/reference/api/pandas.notna.html) **/ notnull()**

* **Description:** A pandas function that returns a same-sized Boolean array indicating whether each value is NOT null (you can also use pd.notnull() as an alias). Note that this function also exists as a DataFrame method.
* **Example:**

**import numpy as np**

**import pandas as pd**

**data = [['Mercury', 2440, np.NaN], ['Venus', 6052, np.NaN,], ['Earth', 6371, 1],**

**['Mars', 3390, np.NaN], ['Jupiter', 69911, 80], ['Saturn', 58232, 83],**

**['Uranus', 25362, 27], ['Neptune', 24622, 14]**

**]**

**cols = ['Planet', 'radius\_km', 'moons']**

**df = pd.DataFrame(data, columns=cols)**

print(df)

print()

pd.notnull(df)

### [**df.fillna()**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html)

* **Description:** A DataFrame method that fills in missing values using specified method
* **Example:**

**import numpy as np**

**import pandas as pd**

**df = pd.DataFrame({'animal':['cardinal', 'gecko', 'raven'],**

**'class': ['Aves', 'Reptilia', 'Aves'],**

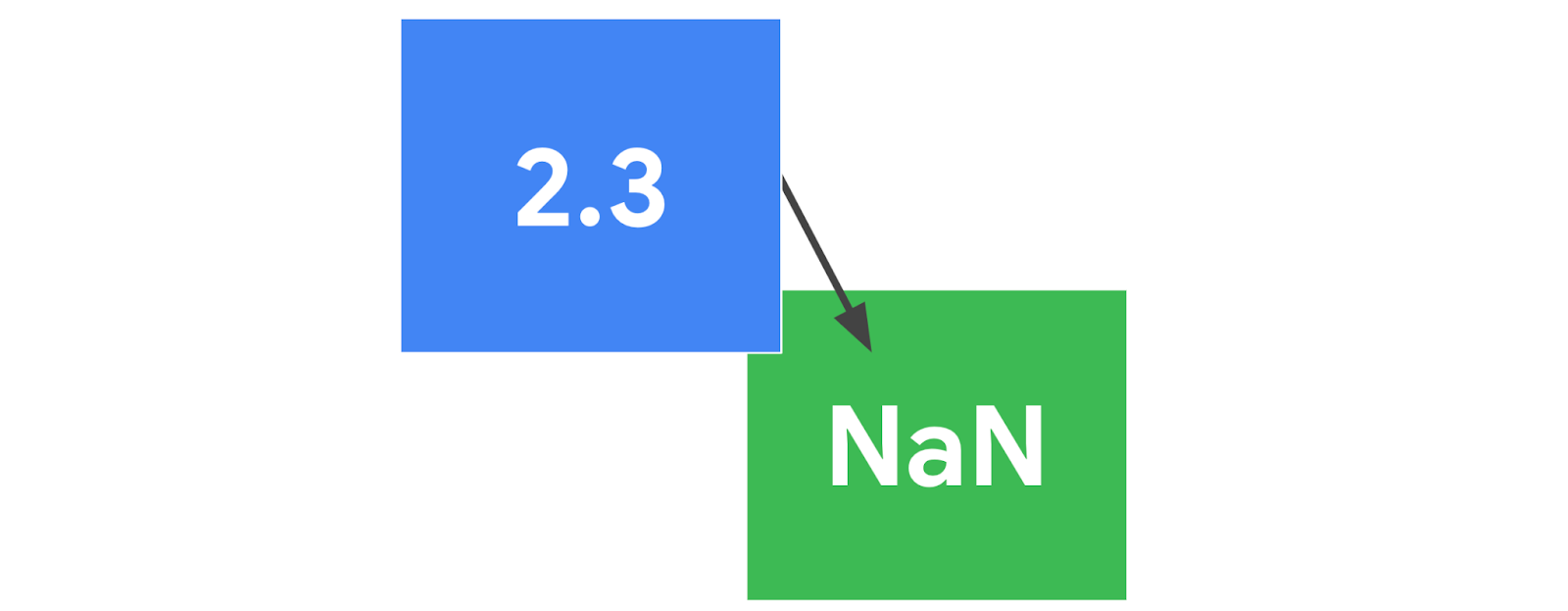
**'color': ['red', 'green', 'black'],**

**'legs': [np.NaN, 4, np.NaN]})**

**print(df)**

**print()**

**df.fillna(2)**

****

### [**df.replace()**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.replace.html)

* **Description:** A DataFrame method that replaces specified values with other specified values. Can also be applied to pandas Series.
* **Example:**

import pandas as pd

df = pd.DataFrame({'animal':['cardinal', 'gecko', 'raven'],

'class': ['Aves', 'Reptilia', 'Aves'],

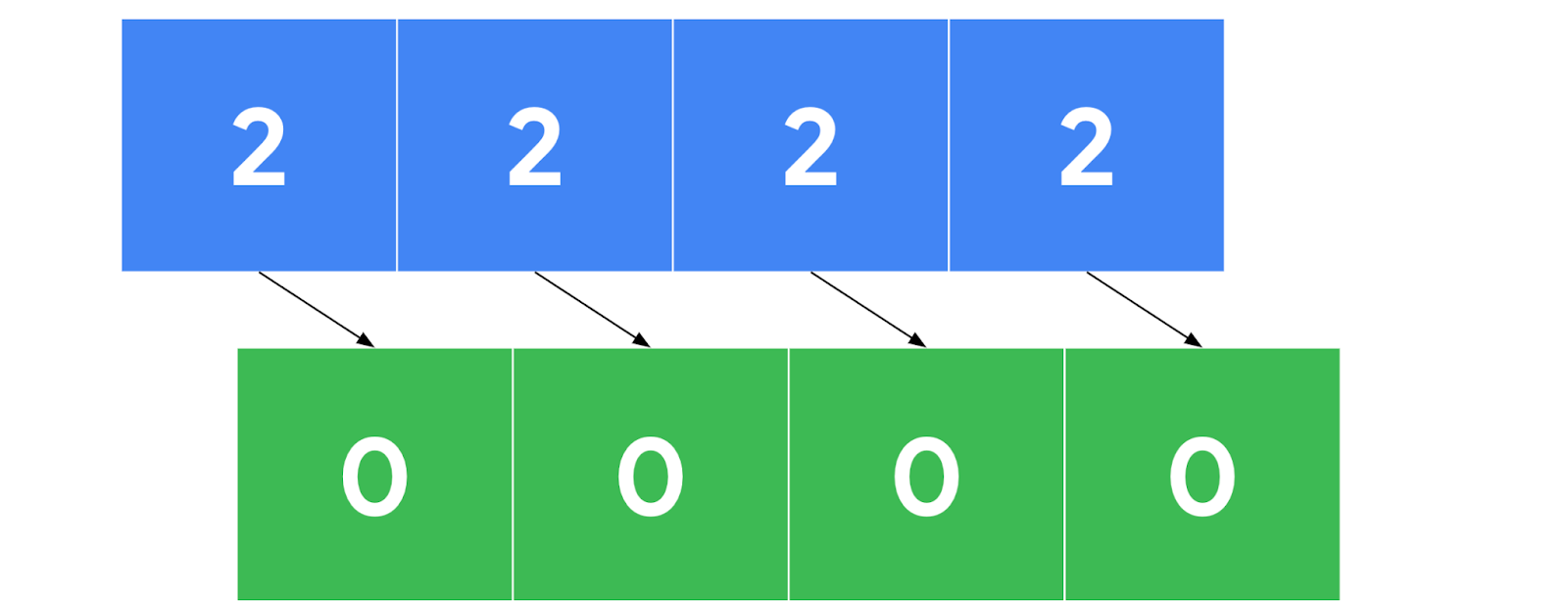
'color': ['red', 'green', 'black'],

'legs': [2, 4, 2]})

print(df)

print()

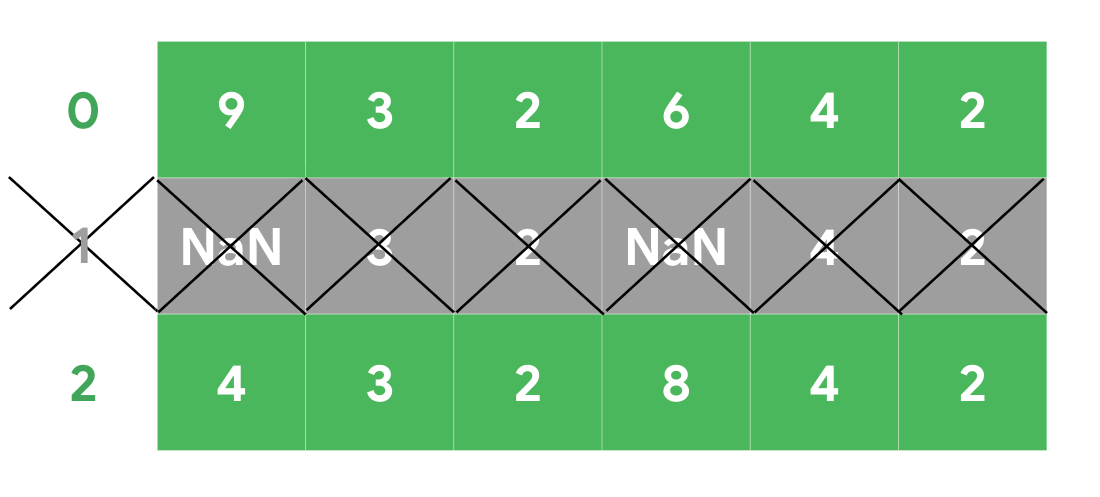
df.replace('Aves', 'bird')

~~~~

### [**df.dropna()**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html)

* **Description:** A DataFrame method that removes rows or columns that contain missing values, depending on the axis you specify.
* **Example:**

df.dropna(axis=0)



## Outliers

The following tools are helpful when dealing with outliers in a dataset.

### [**df.describe()**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html)

* **Description:** A DataFrame method that returns general statistics about the dataframe which can help determine outliers
* **Example:**

df.describe()

count 3.0

mean 2.0

std 1.0

min 1.0

25% 1.5

50% 2.0

75% 2.5

max 3.0

|  | **radius\_km** | **moons** |
| --- | --- | --- |
| **count** | 8.000000 | 8.00000 |
| **mean** | 24547.500000 | 25.87500 |
| **std** | 26191.633528 | 35.58265 |
| **min** | 2440.000000 | 0.00000 |
| **25%** | 5386.500000 | 0.75000 |
| **50%** | 15496.500000 | 8.00000 |
| **75%** | 33579.500000 | 40.25000 |
| **max** | 69911.000000 | 83.00000 |

### [**sns.boxplot()**](https://seaborn.pydata.org/generated/seaborn.boxplot.html)

* **Description:** A seaborn function that generates a box plot. Data points beyond 1.5x the interquartile range are considered outliers.
* **Example:**

## An example graph of a box plot with min, max, lower and upper quartiles, and the median labeled

## 

## Label encoding

The following tools are helpful when performing label encoding.

### [**df.astype()**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.astype.html)

* **Description:** A DataFrame method that allows you to encode its data as a specified dtype. Note that this method can also be used on Series objects.
* **Example:**
* **df = pd.DataFrame({'animal':['cardinal', 'gecko', 'raven'],**
* **'class': ['Aves', 'Reptilia', 'Aves'],**
* **'color': ['red', 'green', 'black'],**
* **'legs': [2, 4, 2]})**
* **df**

**df.dtypes**

**df[['class', 'color']] = df[['class', 'color']].astype('category')**

**df.dtypes**

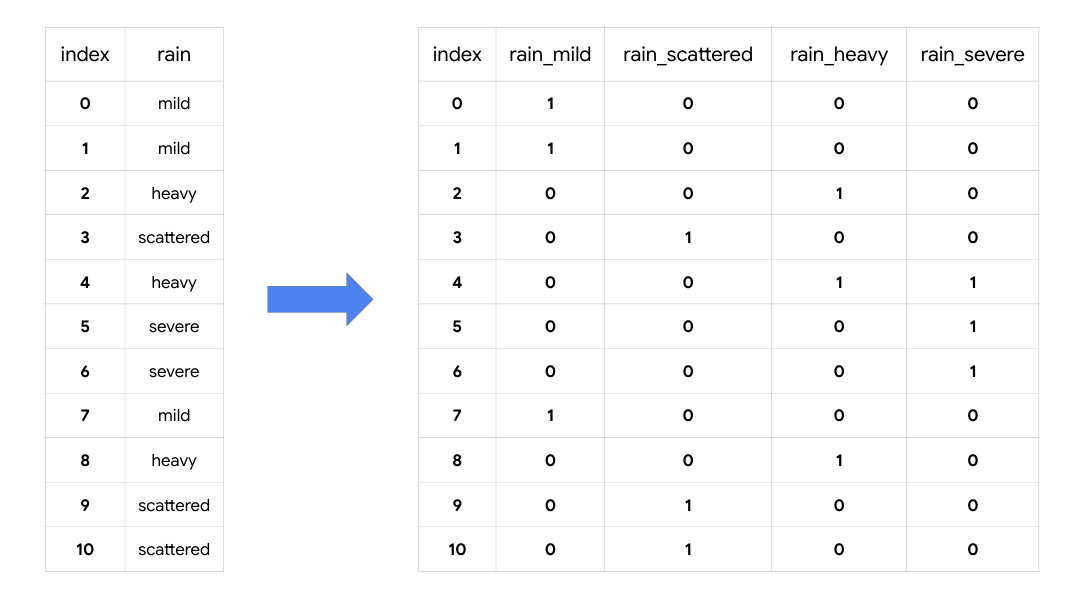
### [**Series.cat.codes**](https://pandas.pydata.org/docs/reference/api/pandas.Series.cat.codes.html)

* **Description:** A Series attribute that returns the numeric category codes of the series
* **Example:**

df['class'].cat.codes

### [**get\_dummies()**](https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html)

* **Description:** Converts categorical values into new binary columns—one for each different category
* **Example:**



### [**LabelEncoder()**](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html)

* **Description:** A transformer from scikit-learn.preprocessing that encodes specified categories or labels with numeric codes. Note that when building predictive models it should only be used on target variables (i.e., *y* data).
* **Example:**

**It can be used to normalize labels:**

**>>> from** **sklearn** **import** preprocessing

**>>>** le = preprocessing.LabelEncoder()

**>>>** le.fit([1, 2, 2, 6])

LabelEncoder()

**>>>** le.classes\_

array([1, 2, 6])

**>>>** le.transform([1, 1, 2, 6])

array([0, 0, 1, 2]...)

**>>>** le.inverse\_transform([0, 0, 1, 2])

array([1, 1, 2, 6])

**It can be used to convert categorical labels into numeric:**

**>>>** le = preprocessing.LabelEncoder()

**>>>** le.fit(["paris", "paris", "tokyo", "amsterdam"])

LabelEncoder()

**>>>** list(le.classes\_)

['amsterdam', 'paris', 'tokyo']

**>>>** le.transform(["tokyo", "tokyo", "paris"])

array([2, 2, 1]...)

**>>>** list(le.inverse\_transform([2, 2, 1]))

['tokyo', 'tokyo', 'paris']

## Key takeaways

There are many tools that data professionals can use to perform data cleaning on a wide range of data. The information you learn from missing data, outliers, and transforming categorical to numeric data will help you prepare datasets for further analysis throughout your career.