



Unit 37.

# Data Tidying

### ● Learning objectives

- ✓ Learners will be able to check for missing data within a given data frame.
- ✓ Learners will be able to delete or replace missing data.
- ✓ Learners will be able to perform descriptive statistics such as mean, median, mode, variance, and standard deviation for data frames.

### ● Learning overview

- ✓ Differentiate situations that require data tidying
- ✓ Search and check missing data
- ✓ Delete missing data or replace it with another value
- ✓ Learn the basics of descriptive statistics such as mean, median, mode, variance, standard deviation, and correlation coefficient
- ✓ Data visualization: learning box about box plots, histograms, and scatter plots

### ● Concepts you will need to know from previous units

- ✓ Know how to select and slice data elements in a data frame
- ✓ Know the basics of Matplotlib visualization

# Keywords

NaN

Tidy data

seaborn

Descriptive  
Statistics

Boxplot

UNIT  
37.

## Data Tidying



# | Mission

## Descriptive Statistics and Visualization of Student Grades

I The University of California, Irvine provides a sample dataset for data learning.


I Among these datasets, download and unzip **student.zip** from <https://archive.ics.uci.edu/ml/datasets/student%2Bperformance>

- ▶ The student-mat.csv file is used as the target data for data analysis.
- ▶ It is converted into a data frame using Pandas.
- ▶ Identify the characteristics of the data.
- ▶ If checking for missing data, run data pre-processing.
- ▶ Calculate the mean, median, and mode.
- ▶ Visualize in a box plot graph.
- ▶ Visualize all the variables in a histogram and scatterplot.

[About](#) [Citation Policy](#) [Donate a Data Set](#) [Contact](#)

☒ Repository ☐ Web










## Machine Learning Repository

Center for Machine Learning and Intelligent Systems

[View ALL Data Sets](#)

Browse Through: **588 Data Sets** Table View [List View](#)

Default Task	Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
<a href="#">Classification (442)</a> <a href="#">Regression (137)</a> <a href="#">Clustering (117)</a> <a href="#">Other (56)</a>	 <a href="#">Abalone</a>	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
<b>Attribute Type</b> <a href="#">Categorical (38)</a> <a href="#">Numerical (396)</a> <a href="#">Mixed (55)</a>	 <a href="#">Adult</a>	Multivariate	Classification	Categorical, Integer	48842	14	1996
<b>Data Type</b> <a href="#">Multivariate (456)</a> <a href="#">Univariate (27)</a> <a href="#">Sequential (57)</a> <a href="#">Time-Series (121)</a> <a href="#">Text (66)</a> <a href="#">Domain-Theory (23)</a> <a href="#">Other (21)</a>	 <a href="#">Annealing</a>	Multivariate	Classification	Categorical, Integer, Real	798	38	
<b>Area</b> <a href="#">Life Sciences (138)</a> <a href="#">Physical Sciences (57)</a> <a href="#">CS / Engineering (215)</a> <a href="#">Social Sciences (38)</a> <a href="#">Business (44)</a> <a href="#">Game (11)</a> <a href="#">Other (80)</a>	 <a href="#">Anonymous Microsoft Web Data</a>		Recommender-Systems	Categorical	37711	294	1998
<b># Attributes</b> <a href="#">Less than 10 (151)</a> <a href="#">10 to 100 (266)</a> <a href="#">Greater than 100 (106)</a>	 <a href="#">Arrhythmia</a>	Multivariate	Classification	Categorical, Integer, Real	452	279	1998
	 <a href="#">Artificial Characters</a>	Multivariate	Classification	Categorical, Integer, Real	6000	7	1992
	 <a href="#">Audiology (Original)</a>	Multivariate	Classification	Categorical	226		1987

<https://archive.ics.uci.edu/ml/datasets.php>

# | Key concept



## 1. Why is Data Tidying Necessary?

- So far, we have learned to create data frames and series, or to bring in other files and turn them into data frames. In fact, the most frequently encountered situation when doing such similar tasks in the field is when the data form of the analysis target is abnormal.
- Abnormal data refers to cases where data is missing, units do not match, or are overlapped without rules. The process of organizing the abnormal data is called data tidying.
- This term was coined by Hadley Wickham(<http://hadley.nz/>) in his paper "Tidy Data"(Paper Link: <https://vita.had.co.nz/papers/tidy-data.html>). The paper is available for download, so please check it out.



## 1. Why is Data Tidying Necessary?

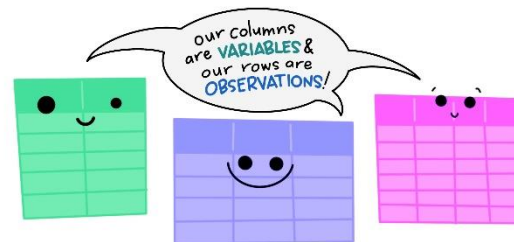
There is something that data scientists in the field always say.

**"80% of data analysis is spent on tidying and preparing data".**

In addition, the reason why this work is so tedious and repetitive is because data preparation and tidying do not end in a single process, but rather, requires repeatedly tidying new problems that are constantly found in the process.

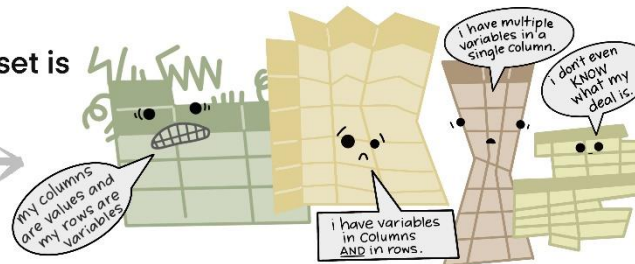
Therefore, Tidy Data provides a standard method of configuring data values within a standardized data set. The standard makes it easy to clean up initial data because you don't have to start from scratch and perform new tasks every time

The standard structure of tidy data means that  
"tidy datasets are all alike..."



"...but every messy dataset is messy in its own way."

—HADLEY WICKHAM



<https://cfss.uchicago.edu/notes/tidy-data/>

## 2. Definition of Tidy Data

“Tidy data is a standard way of mapping the meaning of a dataset to its structure.”

- Hadley Wickham

### I The 3 rules of Tidy Data

- ▶ Each variable must have its own column.
- ▶ Each observation must have its own row.
- ▶ Each value must have its own cell.

country	year	cases	population
Afghanistan	1999	745	15987071
Afghanistan	2000	2666	20995360
Brazil	1999	37737	172006362
Brazil	2000	80488	174604898
China	1999	210258	1272911272
China	2000	210766	128042583

variables

country	year	cases	population
Afghanistan	1999	745	15987071
Afghanistan	2000	2666	20995360
Brazil	1999	37737	172006362
Brazil	2000	80488	174604898
China	1999	210258	1272911272
China	2000	210766	128042583

observations

country	year	cases	population
Afghanistan	1999	745	15987071
Afghanistan	2000	2666	20995360
Brazil	1999	37737	172006362
Brazil	2000	80488	174604898
China	1999	210258	1272911272
China	2000	210766	128042583

values

<https://cfss.uchicago.edu/notes/tidy-data/>

## 3. Messy Data

I The following is a summary of realistic situations in which data tidying is necessary.

### Messy Data

- ▶ If the column is not the name of the variable, but the value itself
- ▶ If there is not just one variable in the column, but multiple variables
- ▶ If the variables are stored in both columns and rows (should be stored in columns)
- ▶ If missing data exists
- ▶ If it's not a sample for the desired period of time
- ▶ If quantitative data is needed but the variable is qualitative
- ▶ If the data types of the values are wrong
- ▶ If there is duplication of data

## 4. Setting Up Data for Practice

### seaborn.load\_dataset

- You can call up and use the needed amount of dataset at seaborn's online repository (<https://github.com/mwaskom/seaborn-data>). By accessing this link, you can also see what data sets there are. The small drawback is that Internet connection is required in order to access the repository.
- Since the result value is returned in the Pandas data frame, it is useful for learning purposes

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = sns.load_dataset('titanic', cache=True)
```



#### Line 5

- Brings in the data set of titanic survivors. At this time, check the local cache first and set it to cache=True in order to set it up for use.

## 4. Setting Up Data for Practice

```
seaborn.load_dataset
```

```
1 df.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
1 df.shape
```

```
(891, 15)
```

## 5. Checking for Missing Data

- | In many cases, the value of the element data is omitted in the data frame. In Pandas, which we are learning now, missing data is displayed as NaN(Not a number).
- | The following are the reasons why missing data occurs. There are many other reasons, but only the common cases are summarized here.
  - ▶ If there are no values corresponding to each other in the two data sets joining E
  - ▶ If the data from an external source is incomplete
  - ▶ If the data is missing at the time of collection because it will be filled up later
  - ▶ If the event continues to occur and accumulate despite an error in the value
  - ▶ If the shape of the data is changed due to adding a new column or row (that was not checked during data reshaping)

## 5. Checking for Missing Data

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  -
0   survived    891 non-null    int64
1   pclass      891 non-null    int64
2   sex         891 non-null    object
3   age         714 non-null    float64
4   sibsp       891 non-null    int64
5   parch       891 non-null    int64
6   fare        891 non-null    float64
7   embarked    889 non-null    object
8   class       891 non-null    category
9   who         891 non-null    object
10  adult_male   891 non-null    bool
11  deck         203 non-null    category
12  embark_town  889 non-null    object
13  alive        891 non-null    object
14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```



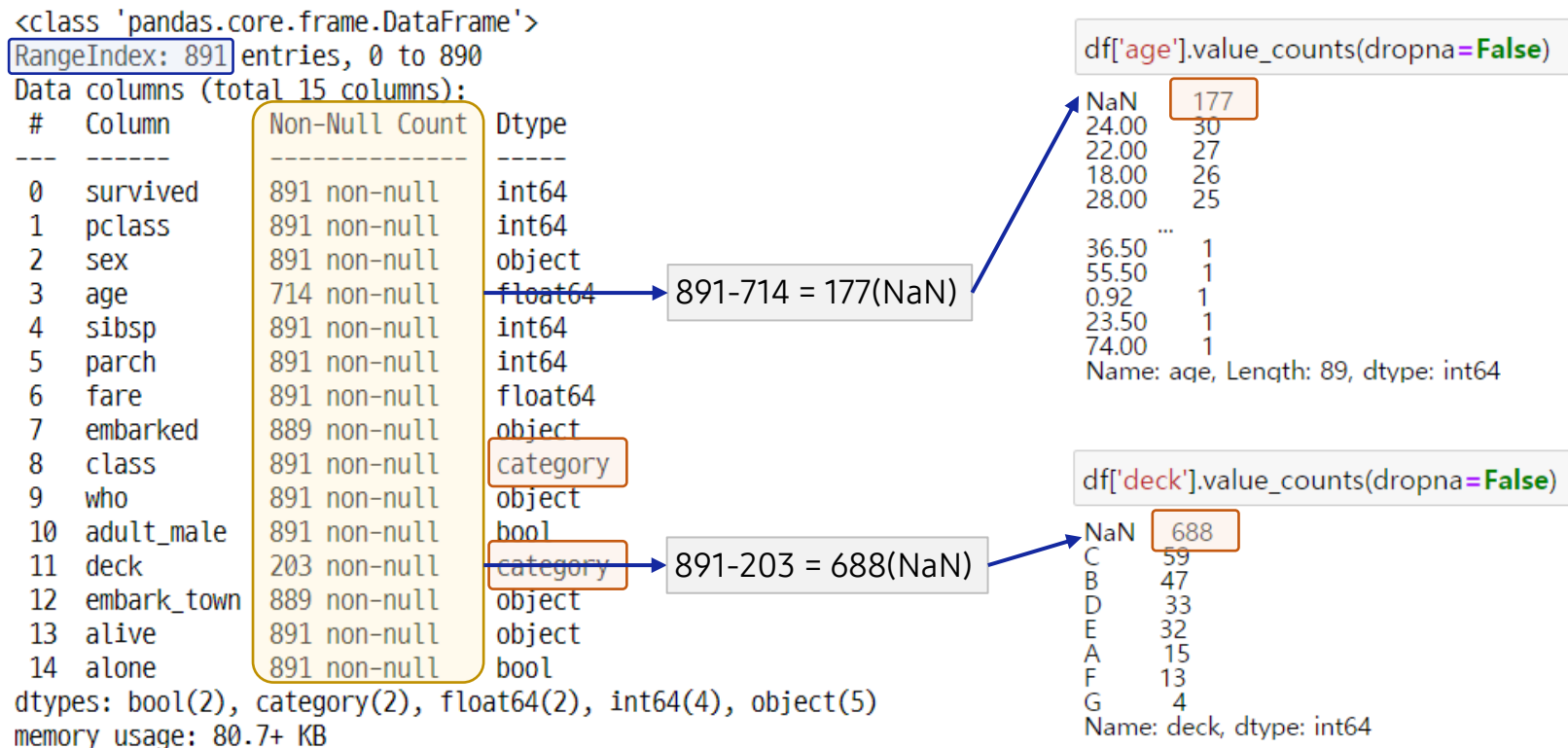
### Line 1

- You can see that the number of valid values among the data held for each column



## 5. Checking for Missing Data

- The Rangeindex shows that each column has 891 data elements. However, it can be seen that the number of non-null data is less than the total amount of data in columns such as age and deck. Simply calculating, it can be confirmed that there are NaN data elements of  $891 - 714 = 177$  for the case of age.



## 5. Checking for Missing Data

If the `dropna=False` parameter is used in the `df.value_counts()` method, the number of missing data can be returned in the form of a Series.

► [https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value\\_counts.html](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value_counts.html)



TIP

- If the parameter is set from `df.vale_counts()` to `dropna=True`, or if the `dropna` parameter is not used at all, the number of data excluding missing data is calculated.

```
1 df['deck'].value_counts(dropna=True)
C    59
B    47
D    33
E    32
A    15
F    13
G     4
Name: deck, dtype: int64
```

## 5. Checking for Missing Data



TIP

- If the parameter is set from `df.value_counts()` to `dropna=True`, or if the `dropna` parameter is not used at all, the number of data excluding missing data is calculated.

```
1 df['deck'].value_counts(dropna=False)
```

```
NaN    688
C       59
B       47
D       33
E       32
A       15
F       13
G        4
Name: deck, dtype: int64
```

# 6. Finding Missing Data

- `isnull()`: Returns True for Missing Data
- `notnull()`: Returns False for Missing Data

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
df.isnull()	0	False	False	False	False	False	False	False	False	False	False	True
NaN → True	1	False	False	False	False	False	False	False	False	False	False	False
	2	False	False	False	False	False	False	False	False	False	False	True
	3	False	False	False	False	False	False	False	False	False	False	False
	4	False	False	False	False	False	False	False	False	False	False	True

## 6. Finding Missing Data

- `isnull()`: Returns True for Missing Data
- `notnull()`: Returns False for Missing Data

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
<code>df.notnull()</code>	0	True	True	True	True	True	True	True	True	True	True	False
<code>NaN → False</code>	1	True	True	True	True	True	True	True	True	True	True	True
	2	True	True	True	True	True	True	True	True	True	True	False
	3	True	True	True	True	True	True	True	True	True	True	True
	4	True	True	True	True	True	True	True	True	True	True	False

## 6. Finding Missing Data

- `isnull()`: Returns True for Missing Data
- `notnull()`: Returns False for Missing Data

```
1 df.isnull()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False
887	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	True	False	False	False	False	False	False	False	True	False	False	False
889	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False

## 6. Finding Missing Data

- `isnull()`: Returns True for Missing Data
- `notnull()`: Returns False for Missing Data

```
1 df.notnull()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
1	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True
887	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
888	True	True	True	False	True	True	True	True	True	True	True	False	True	True	True
889	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
890	True	True	True	True	True	True	True	True	True	True	True	False	True	True	True

## 6. Finding Missing Data

The `.sum()` method treats True and False as 1 and 0. If applied, the total NaN can be obtained

1	<code>df.isnull().sum()</code>			
survived	0	class	0	
pclass	0	who	0	
sex	0	adult_male	0	
age	177	deck	688	
sibsp	0	embark_town	2	
parch	0	alive	0	
fare	0	alone	0	
embarked	2	dtype: int64		

- ▶ If you check the results through this code, there are 177 in the page column.
- ▶ There are two missing data in the `embark_town` column and two in the `deck` column.

### Line 1

- Check how much missing data are for each column in the entire data with numbers.



## 6. Finding Missing Data

```
1 df.isnull().sum().sum()
```

869

 Line 1

- In the previous result series, if `.sum()` is applied once more, we can know the total number of NaN values in the data frame.

```
1 (len(df)-df.count()).sum()
```

869

 Line 1

- `df.count()` returns the number of values other than missing data for each column. The number of missing data can be obtained by subtracting this from the total amount of data.

## 7. Deleting Missing Data

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

I Delete the column.

- ▶ The total number of passengers is 891, and 688 do not have their deck information.
- ▶ Since the proportion of missing data is very high, it can be said that the column is meaningless from the standpoint of processing and analyzing data. In this case, the most common way to deal with missing data is to delete columns with missing data.

## 7. Deleting Missing Data

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

I Delete the column.

```
1 new_df = df.dropna(axis = 1, thresh = 500) :
2
3 new_df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no	True

 Line 1, 3

- 1: thresh = 500 is a command to delete all columns with more than 500 NaN values.
- 3: Since the deck column shows 688 NaNs and there are more than 500 NaNs, we can confirm that all the results are deleted.

## 7. Deleting Missing Data

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

Delete the row.

- ▶ 177 out of 891 have no data on age. If the passenger's age is considered an important variable in data analysis, it is recommended to delete the passenger's data(row) without age data.
- ▶ If subset='age,' delete all rows with NaN values (axis = 0) from the rows in the age column.
- ▶ how = 'all' is deleted only if all data is NaN.

```
1 age_df = df.dropna(axis = 0, how= 'any', subset =['age'])  
2 len(age_df)
```

714

 Line 1, 2

- 1: There were 177 rows without age data above, We deleted this,
- 2: Therefore, it is only normal that the number of the result data is 714.

## 7. Deleting Missing Data

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

1 Delete the column.

```
1 no = df.dropna()  
2 no.isnull().sum()
```

```
survived      0      class      0  
pclass        0      who        0  
sex           0      adult_male  0  
age           0      deck        0  
sibsp         0      embark_town 0  
parch         0      alive        0  
fare          0      alone        0  
embarked      0      dtype: int64
```



### Line 1

- If there is any NaN in the column, delete it.

## 8. Replacing Missing Data with Other

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

- | Just because there is missing data in the preprocessing stage of the data set, deleting all the columns or rows is unfavorable because it reduces the number of data sets to be analyzed.
- | However, replacing NaN's data with values such as 0 or 1 will affect data analysis.
- | Therefore, it is usually a value that replaces missing data, and the mean value and mode value, which represent the distribution and characteristics of the data set, are obtained and filled.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 1) Replacing with the Mean

- ▶ <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mean.html?highlight=mean#pandas.DataFrame.mean>

```
1 import pandas as pd
2 import seaborn as sns
3
4 df = sns.load_dataset('titanic', cache=True)
5
6 df['age'].head(10)
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
Name: age, dtype: float64
```

## 8. Replacing Missing Data with Other Data

```
1 import pandas as pd
2 import seaborn as sns
3
4 df = sns.load_dataset('titanic', cache=True)
5
6 df['age'].head(10)
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
Name: age, dtype: float64
```



### Line 4, 6

- 4: Bring up the Titanic Dataset.
- 6: Check the data in the age column, the data is found as NaN.



## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 1) Replacing with the Mean

```
1 avg_age = df['age'].mean(axis=0) ,  
2 avg_age
```

29.69911764705882



#### Line 1

- The mean of the data age is stored in avg\_age. It is the mean of the data in the age column.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 1) Replacing with the Mean

```
1 median_age = df['age'].median(axis=0)
2
3
4 median_age
```

29.69911764705882

#### Line 1, 2

- 1: The median, using the median() method, can also be used as replacement instead of the mean.
- 2: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.median.html?highlight=median#pandas.DataFrame.median>

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 1) Replacing with the Mean

```
1 df['age'].fillna(avg_age, inplace=True)
```

 Line 1

- NaN data elements are substituted with the mean using fillna(). Let's replace it using the median value median\_age, calculated earlier.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 1) Replacing with the Mean

```
1 df['age'].head(10)
```

0	22.000000
1	38.000000
2	26.000000
3	35.000000
4	35.000000
5	29.699118
6	54.000000
7	2.000000
8	27.000000
9	14.000000

Name: age, dtype: float64

 Line 1

- We can see that the missing data is replaced with the mean.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 1) Replacing with the Mean

```
1 df.isnull().sum()
```

survived	0	class	0
pclass	0	who	0
sex	0	adult_male	0
age	177	deck	688
sibsp	0	embark_town	2
parch	0	alive	0
fare	0	alone	0
embarked	2	dtype: int64	

 Line 1

- It appears that there is no NaN replaced in the age column.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

2) Replacing with the Maximum Value.

- ▶ [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.value\\_counts.html?highlight=value\\_counts](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.value_counts.html?highlight=value_counts)
- ▶ <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.idxmax.html?highlight=idxmax#pandas.DataFrame.idxmax>
- ▶ Let's search for the missing data of embark\_town by searching for the name of the city with the most passengers and replace it with the data.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 2) Replacing with the Maximum Value

```
1 import pandas as pd
2 import seaborn as sns
3
4 df = sns.load_dataset('titanic', cache=True)
5
```

#### Line 4

- Bring up the Titanic data set.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 2) Replacing with the Maximum Value

```
1 town_count = df['embark_town'].value_counts(dropna=True)
2
3 town_count
```

```
Southampton    644
Cherbourg       168
Queenstown      77
Name: embark_town, dtype: int64
```

 Line 1

- Return the time series including unique rows in the corresponding column. Exclude missing data.



## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 2) Replacing with the Maximum Value

```
1 most=town_count.idxmax()  
2  
3 most
```

'Southampton'

 Line 1

- `df.idxmax()` Return the index in which the maximum value first occurs on the requested axis.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 2) Replacing with the Maximum Value

```
1 df['embark_town'].fillna(most, inplace=True)
2
3 df.isnull().sum()
```

survived	0	class	0
pclass	0	who	0
sex	0	adult_male	0
age	177	deck	688
sibsp	0	embark_town	0
parch	0	alive	0
fare	0	alone	0
embarked	2	dtype: int64	

#### Line 1, 3

- 1: Using the fillna() method, NaN data elements are substituted with the names of the most embarking towns stored in the variable most.
- 3: Missing data in the embark\_town column has been replaced by Southampton, resulting in no missing data in the column.

## 8. Replacing Missing Data with Other Data

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

### 3) Replacing with Nearest Neighbor Values

- ▶ This is not true for all data, but usually due to its nature, neighboring data often have similarities. After checking the characteristics of the dataset, the missing data is replaced with previous data or the one immediately after.

- Changing the Entire Row Index : `dataframeobjectname.index = array of row indices that are to be changed`
- Changing the Entire Column Name: `datagrameobject.columns – array of column names that are to be changed`

```
1 df_01=df['embark_town'].fillna(method = 'ffill', inplace=True)
```

```
1 df_02=df['embark_town'].fillna(method = 'bfill', inplace=True)
```

- ▶ Print df\_01 and df\_02 and compare the two data frames to see how the place where the missing data was became filled!

| Let's code

## Step 1

### I Preparing Data and Creating Data Frame

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 student_data = pd.read_csv("./data/student/student-mat.csv")
7 student_data.head()

```

	school;sex;age;address;famsize;Pstatus;Medu;Fedu;Mjob;Fjob;reason;guardian;traveltime;studytime;failures;schoolsup;famsup;paid;activities;nursery;higher;internet;romantic;famrel;freetime;goout;Dalc;Walc;health;absences;G1;G2;G3
0	GP;"F";18;"U";"GT3";"A";4;4;"at_home";"teacher...
1	GP;"F";17;"U";"GT3";"T";1;1;"at_home";"other";...
2	GP;"F";15;"U";"LE3";"T";1;1;"at_home";"other";...
3	GP;"F";15;"U";"GT3";"T";4;2;"health";"services...
4	GP;"F";16;"U";"GT3";"T";3;3;"other";"other";"h...

- ▶ Although the data input has been confirmed, the data is difficult to handle in this state. It can be confirmed that the downloaded data is divided into ' ; '.
- ▶ Usually, csv files are divided by ' , ', but this file is divided into semicolons, make it difficult to check visually.
- ▶ In order to change the character symbol that separates the data, the parameter sep= 'separating character symbol' is used to designate it

# Step 1

## I Preparing Data and Creating Data Frame

```
1 student_data = pd.read_csv("./data/student/student-mat.csv", sep=';')
2 student_data.head()
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10

5 rows × 33 columns

## Step 1

## | Checking Data Characteristics

```
1 student_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 395 entries, 0 to 394
```

```
Data columns (total 33 columns):
```

#	Column	Non-Null Count	Dtype
0	school	395 non-null	object
1	sex	395 non-null	object
2	age	395 non-null	int64
3	address	395 non-null	object
4	famsize	395 non-null	object
5	Pstatus	395 non-null	object
6	Medu	395 non-null	int64
7	Fedu	395 non-null	int64
8	Mjob	395 non-null	object
9	Fjob	395 non-null	object
10	reason	395 non-null	object
11	guardian	395 non-null	object
12	traveltime	395 non-null	int64
13	studytime	395 non-null	int64
14	failures	395 non-null	int64
15	schoolsup	395 non-null	object
16	famsup	395 non-null	object
17	paid	395 non-null	object
18	activities	395 non-null	object
19	nursery	395 non-null	object
20	higher	395 non-null	object
21	internet	395 non-null	object
22	romantic	395 non-null	object
23	famrel	395 non-null	int64
24	freetime	395 non-null	int64
25	goout	395 non-null	int64
26	Dalc	395 non-null	int64
27	Walc	395 non-null	int64
28	health	395 non-null	int64
29	absences	395 non-null	int64
30	G1	395 non-null	int64
31	G2	395 non-null	int64
32	G3	395 non-null	int64

dtypes: int64(16), object(17)  
memory usage: 102.0+ KB

Line 1

- non-null means that there is no null data

## Step 1

### I Searching for any NaN in the Data

```
1 student_data.isnull().sum()
```

school	0	traveltime	0	freetime	0
sex	0	studytime	0	goout	0
age	0	failures	0	Dalc	0
address	0	schoolsup	0	Walc	0
famsize	0	famsup	0	health	0
Pstatus	0	paid	0	absences	0
Medu	0	activities	0	G1	0
Fedu	0	nursery	0	G2	0
Mjob	0	higher	0	G3	0
Fjob	0	internet	0	dtype: int64	
reason	0	romantic	0		
guardian	0	famrel	0		

 Line 1

- Looking at the results, it can be confirmed that the number of NaN data for each column is 0.



## Step 1

### I Searching for any NaN in the Data

```
1 student_data.isnull().sum().sum()
```

```
0
```



#### Line 1

- Obtain the number of columns with #NaN data

## Step 1

### I Understanding the Meaning of the Columns for Data Comprehension

- ▶ For data analysis, it is fundamental to understand the meaning of each column name of the data frame.
- ▶ In the case of this data, it is helpful because it is written in detail in the study.txt file included in the compressed file. However, on a daily basis, there are often no documents explaining these column names. In this case, it is necessary to contact the person who received the data to check the information on the column of the data set.

# Step 1

## I Understanding the Meaning of the Columns for Data Comprehension

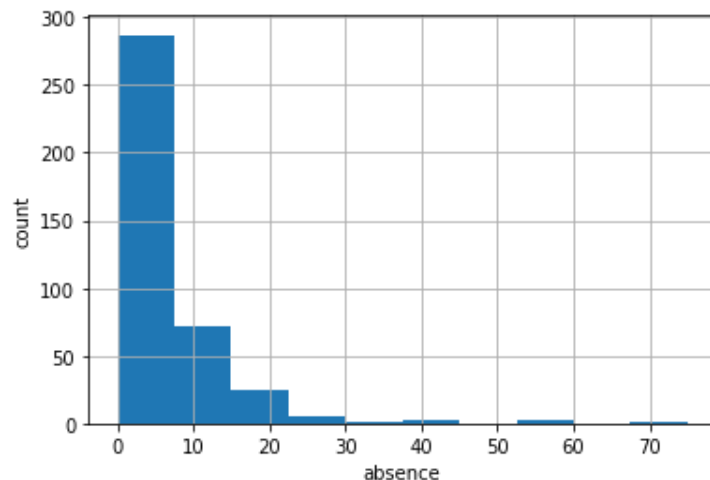
1 school - student's school (binary: "GP" - Gabriel Pereira or "MS" - Mousinho da Silveira)  
2 sex - student's sex (binary: "F" - female or "M" - male)  
3 age - student's age (numeric: from 15 to 22)  
4 address - student's home address type (binary: "U" - urban or "R" - rural)  
5 famsize - family size (binary: "LE3" - less or equal to 3 or "GT3" - greater than 3)  
6 Pstatus - parent's cohabitation status (binary: "T" - living together or "A" - apart)  
7 Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)  
8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)  
9 Mjob - mother's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at\_home" or "other")  
10 Fjob - father's job (nominal: "teacher", "health" care related, civil "services" (e.g. administrative or police), "at\_home" or "other")  
11 reason - reason to choose this school (nominal: close to "home", school "reputation", "course" preference or "other")  
12 guardian - student's guardian (nominal: "mother", "father" or "other")  
13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)  
14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)  
15 failures - number of past class failures (numeric: n if 1 ≤ n ≤ 3, else 4)  
16 schoolsup - extra educational support (binary: yes or no)  
17 famsup - family educational support (binary: yes or no)  
18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)  
19 activities - extra-curricular activities (binary: yes or no)  
20 nursery - attended nursery school (binary: yes or no)  
21 higher - wants to take higher education (binary: yes or no)  
22 internet - Internet access at home (binary: yes or no)  
23 romantic - with a romantic relationship (binary: yes or no)  
24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)  
25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)  
26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)  
27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)  
28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)  
29 health - current health status (numeric: from 1 - very bad to 5 - very good)  
30 absences - number of school absences (numeric: from 0 to 93)

## Step 2

- Let's visualize the number of absent days of students in a histogram. The number of days of absence is the column 30 absences.

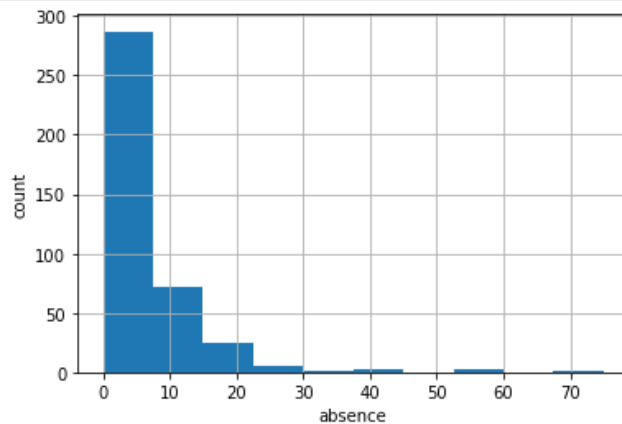
► [https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.hist.html#matplotlib-pyplot-hist](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html#matplotlib-pyplot-hist)

```
1 plt.hist(student_data['absences'])  
2  
3 plt.xlabel('absence')  
4 plt.ylabel('count')  
5  
6 plt.grid(True)  
7
```



## Step 2

```
1 plt.hist(student_data['absences'])  
2  
3 plt.xlabel('absence')  
4 plt.ylabel('count')  
5  
6 plt.grid(True)  
7
```



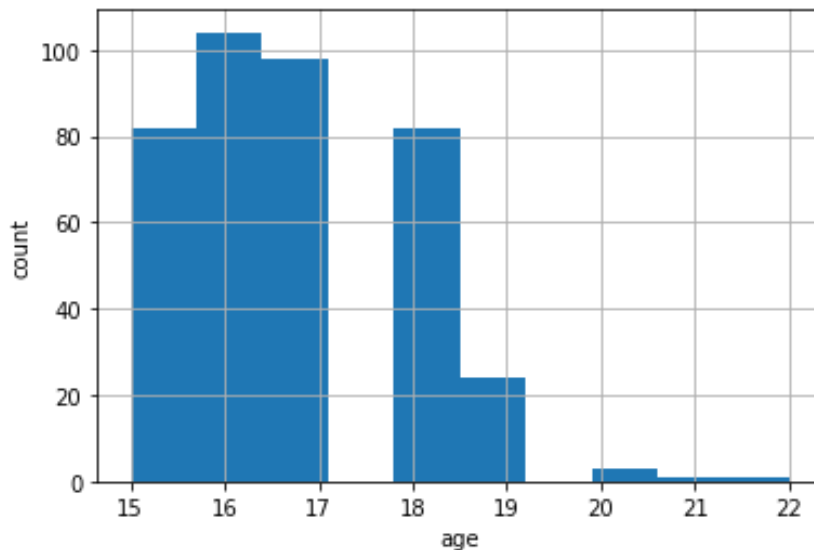
## Line 1, 6

- 1: Specify the variable to be graphed on the histogram
- 6: Add a grid to the graph

## Step 2

Let's make a histogram graph of other variables that have numeric data perform an exploratory data analysis.

```
1 plt.hist(student_data['age'])  
2  
3 plt.xlabel('age')  
4 plt.ylabel('count')  
5  
6 plt.grid(True)
```



## Step 3

- I Find the basic descriptive statistics such as mean, median, mode, variance, and standard deviation etc.
  - ▶ First of all, we learned in the previous lessons that we can check the results of various and basic descriptive statistics of the corresponding data frame using Pandas' describe() method.

```
1 student_data.describe()
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
<b>count</b>	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000
<b>mean</b>	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.944304	3.235443	3.108861	1.481013	2.291139	3.554430	5.708861	10.908861	10.713924	10.415190
<b>std</b>	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.896659	0.998862	1.113278	0.890741	1.287897	1.390303	8.003096	3.319195	3.761505	4.581443
<b>min</b>	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	3.000000	0.000000	0.000000
<b>25%</b>	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000	1.000000	1.000000	3.000000	0.000000	8.000000	9.000000	8.000000
<b>50%</b>	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000	1.000000	2.000000	4.000000	4.000000	11.000000	11.000000	11.000000
<b>75%</b>	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000	2.000000	3.000000	5.000000	8.000000	13.000000	13.000000	14.000000
<b>max</b>	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	75.000000	19.000000	19.000000	20.000000

## Step 3

### I Median

- ▶ The median refers to the middle value of a data that is rearranged in the order of size.
- ▶ Assuming that the median value of studytime is obtained,

```
1 student_data['studytime'].median()
```

```
2.0
```



## Step 3

### I Mode

- ▶ The mode is the most frequent value in the data.
- ▶ Assuming that we obtain the mode of studytime

```
1 student_data['studytime'].mode()
```

```
0    2  
dtype: int64
```

## Step 3

### I Variance

- ▶ It is possible to check whether the data is scattered or concentrated around the mean by calculating the variance. After designating the reference variable, the `var()` method is used.
- ▶ Square the observed value minus the average, add it all, and divide it by the total number. That is, it's the sum of all squared differences. If you add all the deviations minus the mean from the observed values, you get zero, so you add them in squares.

```
1 student_data['studytime'].var()
```

```
0.704324359056738
```

 Line 1

- The smaller the result value, the smaller the degree of scattering the data.

## Step 3

### I Standard Deviation

- ▶ Where there is a lot of data, the average is often used as the value that represents the data. The standard deviation, one of the scatter plots, is a representative figure indicating how spread out the data is around the average. The unit of the standard deviation is identical with the unit of data, unlike the variance which uses the square root. If the standard deviation is close to the center, it means that the data values are concentrated near the average. The larger the standard deviation, the more widespread the data values are.

```
1 student_data['studytime'].std()
```

```
0.839240346418556
```

 Line 1

- The smaller the result value, the smaller the degree of scatter in the data.

## Step 3

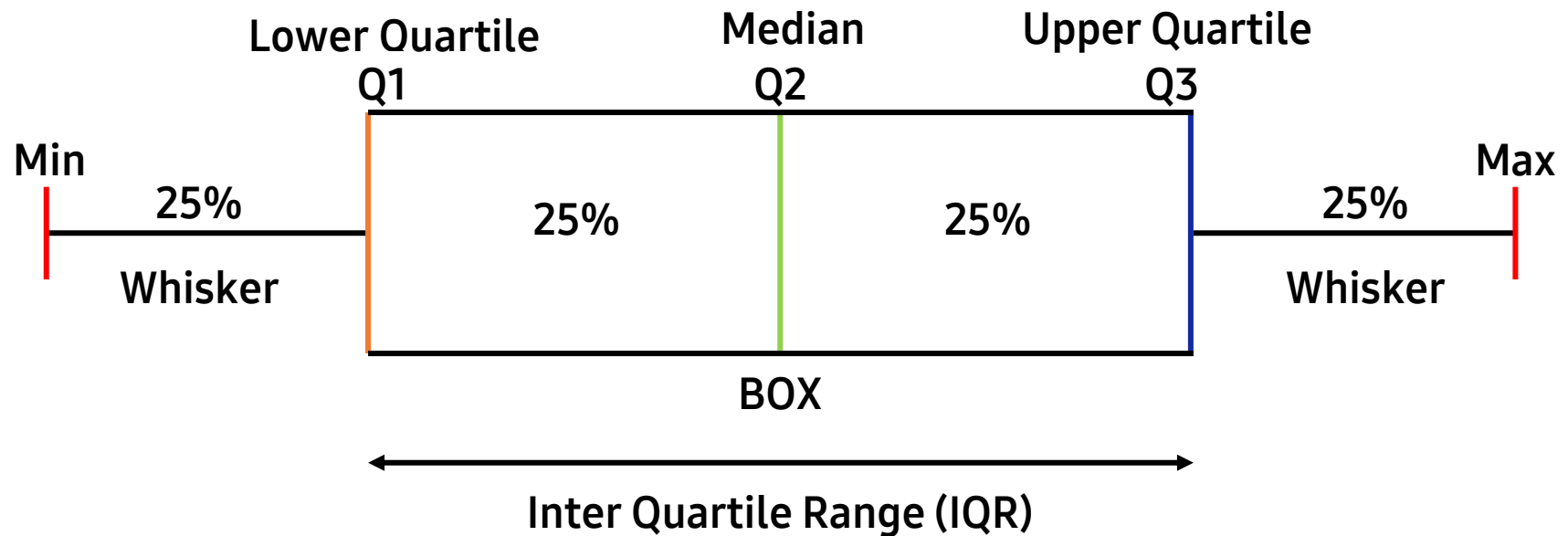
### I Visualization through Box Plot

#### ▸ Necessary Concepts for Understanding Box Plots

<b>Percentile</b>	Data divided into hundred equal parts
<b>Quartile</b>	Data divided into four parts
<b>Median Value(Q2)</b>	The value at the middle of the data (Half of the observations are greater than or equal, and the other half are smaller or equal)
<b>The 3<sup>rd</sup> (Upper) Quartile (Q3)</b>	The median of the top 50% based on the median value. The value corresponding to the top 25% of the entire data.
<b>The 1<sup>st</sup> (Lower) Quartile (Q1)</b>	The median of the bottom 50% based on the median value. The value corresponding to the bottom 25%, that is 75% of the total data.

## Step 3

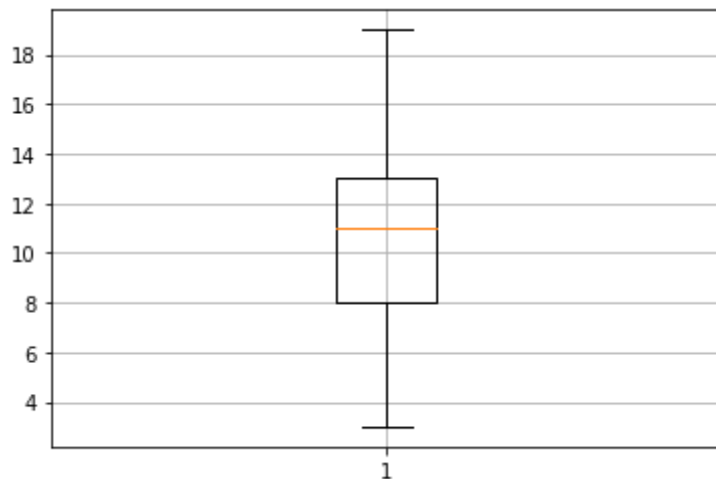
| Visualization through Box Plot



## Step 3

### Visualization through Box Plot

```
1 plt.boxplot(student_data['G1'])  
2 plt.grid(True)
```



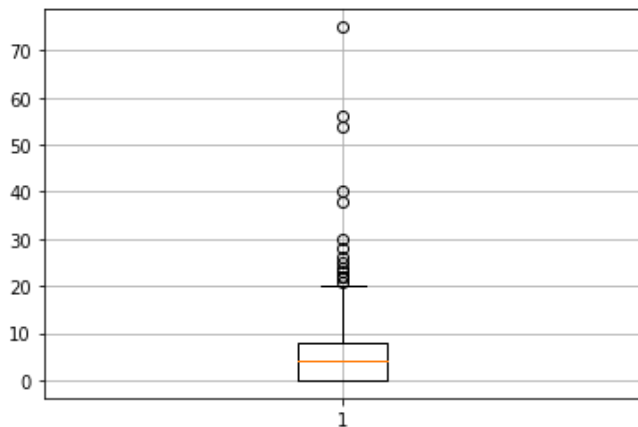
#### Line 1

- 1<sup>st</sup> Semester Grades

## Step 3

### Visualization through Box Plot

```
1 plt.boxplot(student_data['absences'])  
2 plt.grid(True)
```



#### Line 1

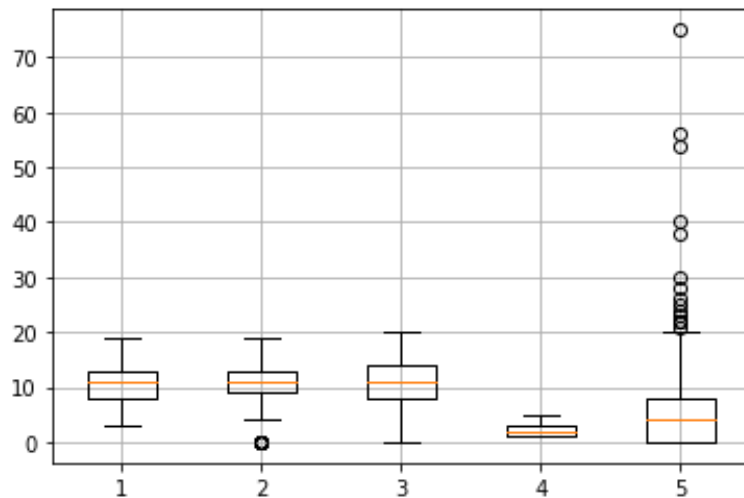
- Number of absent days

- ▶ Looking at the graph, it can be seen that there are many abnormal data in the case of the box plot of the number of absences.

## Step 3

### Visualization through Box Plot

```
1 plt.boxplot([student_data['G1'],student_data['G2'],student_data['G3'],student_data['walc'],student_data['absences']])  
2 plt.grid(True)
```



- ▶ If you look at the box plot of the first semester grades, the second semester grades, the third semester grades, the weekend night outs statistics, and the number of days of absence, you can get insights on student performance.



## Step 4

### I Coefficient of Variation (CV)

- ▶ Data on variables with different units of measurement cannot be compared simply. This is because if the size of the data is different, the deviation tends to increase when the measurement unit is large.

**Ex** The standard deviation between stock prices and gas prices cannot be compared simply.

- ▶ What we can use in these situations is the coefficient of the variation.
- ▶ It is the value of the standard deviation divided by the mean. This can be used to compare data of different specifications regardless of size.

**Coefficient of Variation (CV) = Standard Deviation/Mean**

## Step 4

### I Coefficient of Variation (CV)

```
1 study_time_cv= student_data['studytime'].std() / student_data['studytime'].mean()  
2 print(study_time_cv)
```

```
0.412313354272798
```

```
1 absences_cv= student_data['absences'].std() / student_data['absences'].mean()  
2 print(absences_cv)
```

```
1.4018726369879067
```

## Step 4

### I Coefficient of Variation (CV)

- ▶ The describe() method does not show the result of the coefficient of variation for the whole, It can be applied as follows.
- ▶ In order to find the coefficient of variation for the whole, it can be applied as follows.
- ▶ However, it should be noted that if the mean to be compared is 0 or close to 0, the coefficient of variation may be infinitely large.

## Step 4

### I Coefficient of Variation (CV)

```
1 cv = student_data.std()/student_data.mean()
2 cv
```

```
<ipython-input-160-52c8c8259d45>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
```

```
cv = student_data.std()/student_data.mean()
```

age	0.076427	Dalc	0.601441
Medu	0.398177	Walc	0.562121
Fedu	0.431565	health	0.391147
traveltime	0.481668	absences	1.401873
studytime	0.412313	G1	0.304266
failures	2.225319	G2	0.351086
famrel	0.227330	G3	0.439881
freetime	0.308725	dtype: float64	
goout	0.358098		



#### Line 1

- Return the CV for the entire column as Series

## Step 4

### I Coefficient of Variation (CV)

```
1 cv = student_data.std()/student_data.mean()
2 cv
```

```
<ipython-input-160-52c8c8259d45>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
```

```
cv = student_data.std()/student_data.mean()
```

age	0.076427	Dalc	0.601441
Medu	0.398177	Walc	0.562121
Fedu	0.431565	health	0.391147
traveltime	0.481668	absences	1.401873
studytime	0.412313	G1	0.304266
failures	2.225319	G2	0.351086
famrel	0.227330	G3	0.439881
freetime	0.308725	dtype: float64	
goout	0.358098		



#### Line 2

- However, since the entire data may not be numeric, it is recommended to designate a specific column, as specified in the content of the error message

## Step 4

### I Covariance

- ▶ The covariance represents the relationship between the two variables.
  - If the values of the covariance is positive, the two variables are positive.
  - If the value of the covariance is negative, the two variables are negative.
- ▶ Multiply the deviation between the two variables and calculate it by averaging it. It is used to calculate the variance of two or more variables.
- ▶ It can be calculated using the `cov()` method.  
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.cov.html>

## Step 4

## I Covariance

```
1 student_data.cov(min_periods=None, ddof=1)
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	absences
age	1.628285	-0.228619	-0.226949	0.062873	-0.004434	0.231221	0.061717	0.020947	0.180364	0.149039	0.192733	-0.110326	1.789501
Medu	-0.228619	1.198445	0.742717	-0.131061	0.059667	-0.192681	-0.003842	0.033779	0.078115	0.019341	-0.066440	-0.071349	0.878622
Fedu	-0.226949	0.742717	1.184180	-0.120073	-0.008379	-0.202641	-0.001337	-0.013963	0.052220	0.002313	-0.017702	0.022303	0.213134
traveltime	0.062873	-0.131061	-0.120073	0.486513	-0.059070	0.047844	-0.010512	-0.011861	0.022162	0.085941	0.120478	0.007274	-0.072255
studytime	-0.004434	0.059667	-0.008379	-0.059070	0.704324	-0.108321	0.029898	-0.120041	-0.059706	-0.146533	-0.274304	-0.088228	-0.421127
failures	0.231221	-0.192681	-0.202641	0.047844	-0.108321	0.553017	-0.029564	0.068329	0.103123	0.090118	0.135964	0.068059	0.379265
famrel	0.061717	-0.003842	-0.001337	-0.010512	0.029898	-0.029564	0.803997	0.134974	0.064454	-0.061974	-0.130952	0.117252	-0.318287
freetime	0.020947	0.033779	-0.013963	-0.011861	-0.120041	0.068329	0.134974	0.997725	0.316944	0.185954	0.190163	0.105173	-0.464274
goout	0.180364	0.078115	0.052220	0.022162	-0.059706	0.103123	0.064454	0.316944	1.239388	0.264763	0.602744	-0.014824	0.394718
Dalc	0.149039	0.019341	0.002313	0.085941	-0.146533	0.090118	-0.061974	0.185954	0.264763	0.793420	0.742852	0.095579	0.797758
Walc	0.192733	-0.066440	-0.017702	0.120478	-0.274304	0.135964	-0.130952	0.190163	0.602744	0.742852	1.658678	0.165585	1.404774

Line 1

- Returns the covariance value of each column in the data frame.

## Step 4

### I Covariance

```
1 np.cov(student_data['G1'],student_data['G3'])
```

```
array([[11.01705327, 12.18768232],  
       [12.18768232, 20.9896164 ]])
```

#### Line 1

- The covariance of NumPy can also be calculated through the cov() method. Two series columns, that is, the result of covariance for two series data.
- 
- ▶ Analysis of the matrix above is as follows.
    - The covariance values of G1 and G3: matrix elements (1,2) and (2,1) 12.18768232
    - Variance of G1: matrix element (1,1) 11.01705327
    - Variance of G3: matrix element (2,2) 20.9896164



## Step 4

### I Covariance

```
1 print(student_data['G1'].var())  
2 print(student_data['G3'].var())
```

```
11.017053267364899  
20.989616397866737
```



#### Line 1, 2

- 1: Verification through var() to obtain the variance is the same as the result of the matrix element above
- 2: Verification through var() to obtain the variance is the same as the result of the matrix element above

## Step 4

### I Correlation Coefficient

- ▶ The covariance equation itself depends on the scale and unit of each variable. The correlation coefficient eliminates the dependence of each variable on the scale and finds out the relationship between data.
- ▶ The covariance can tell you what the relationship between the two variables is, but there are cases where a correlation coefficient is needed because the size of the relationship cannot be explained.
- ▶ Simply put, the correlation coefficient measures the degree to which two variables move(?) together.

1.0	Complete positive correlation. This means that when one variable moves to a specific size, the other moves in the same direction at the same rate.
0.0	It means that the two variables have no relationship.
-1.0	Complete negative correlation or inverse correlation. This means that when one variable moves to a specific size, the other moves in the opposite direction.

- ▶ Can be calculated using the `corr()` method.  
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html>

## Step 4

## | Correlation Coefficient

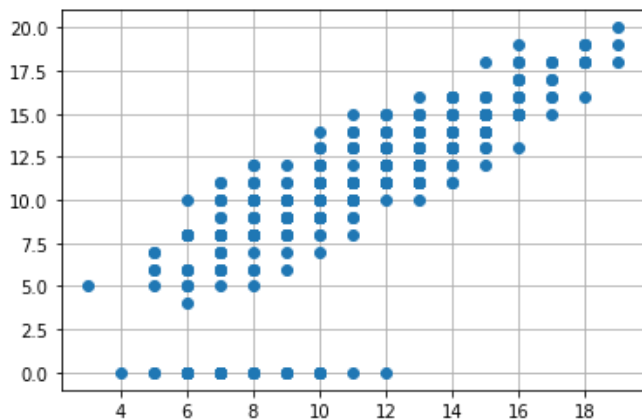
1 student\_data.corr()

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	absences
age	1.000000	-0.163658	-0.163438	0.070641	-0.004140	0.243665	0.053940	0.016434	0.126964	0.131125	0.117276	-0.062187	0.175230
Medu	-0.163658	1.000000	0.623455	-0.171639	0.064944	-0.236680	-0.003914	0.030891	0.064094	0.019834	-0.047123	-0.046878	0.100285
Fedu	-0.163438	0.623455	1.000000	-0.158194	-0.009175	-0.250408	-0.001370	-0.012846	0.043105	0.002386	-0.012631	0.014742	0.024473
traveltime	0.070641	-0.171639	-0.158194	1.000000	-0.100909	0.092239	-0.016808	-0.017025	0.028540	0.138325	0.134116	0.007501	-0.012944
studytime	-0.004140	0.064944	-0.009175	-0.100909	1.000000	-0.173563	0.039731	-0.143198	-0.063904	-0.196019	-0.253785	-0.075616	-0.062700
failures	0.243665	-0.236680	-0.250408	0.092239	-0.173563	1.000000	-0.044337	0.091987	0.124561	0.136047	0.141962	0.065827	0.063726
famrel	0.053940	-0.003914	-0.001370	-0.016808	0.039731	-0.044337	1.000000	0.150701	0.064568	-0.077594	-0.113397	0.094056	-0.044354
freetime	0.016434	0.030891	-0.012846	-0.017025	-0.143198	0.091987	0.150701	1.000000	0.285019	0.209001	0.147822	0.075733	-0.058078
goout	0.126964	0.064094	0.043105	0.028540	-0.063904	0.124561	0.064568	0.285019	1.000000	0.266994	0.420386	-0.009577	0.044302
Dalc	0.131125	0.019834	0.002386	0.138325	-0.196019	0.136047	-0.077594	0.209001	0.266994	1.000000	0.647544	0.077180	0.111908
Walc	0.117276	-0.047123	-0.012631	0.134116	-0.253785	0.141962	-0.113397	0.147822	0.420386	0.647544	1.000000	0.092476	0.136291

## Step 4

### | Scatter Plot

```
1 plt.plot(student_data['G1'],student_data['G3'],'o')  
2 plt.grid(True)
```



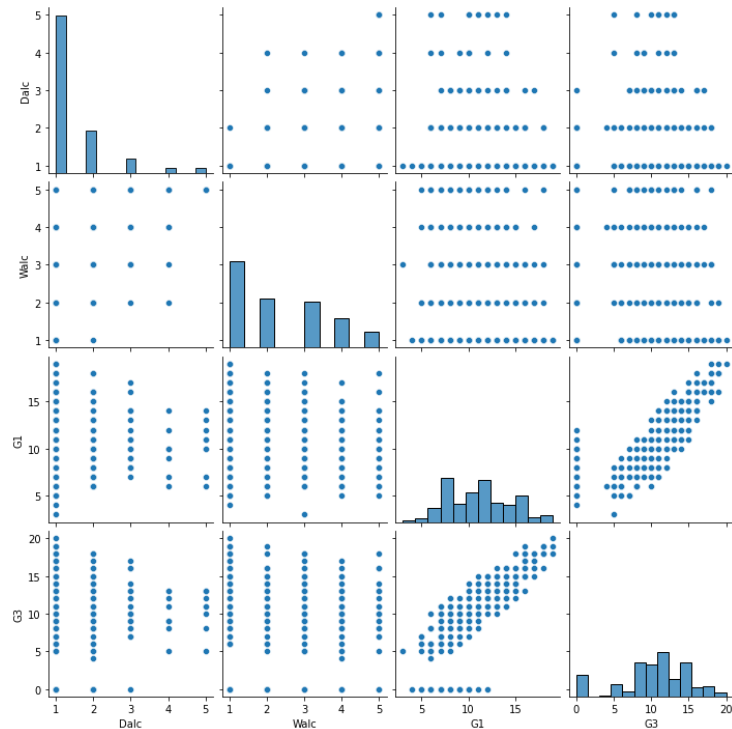
#### Line 1

- The graph displays the comparison results of the first and final tests. It shows that people with good grades from the beginning often perform well until the end.

## Step 5

1 Draw a histogram and scatter plot of the variables you want to compare.

```
1 sns.pairplot(student_data[['Dalc', 'Walc', 'G1', 'G3']])  
2 plt.grid(True)
```



# | Pair programming



## Pair Programming Practice

### Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns “driving the mouse.” Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

### Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a “divide-and-conquer” strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

### Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.



## Pair Programming Practice

### ■ Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

### ■ Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor's question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students "divide and conquer." When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.



**Q1.** After converting the `universities_ranking.csv` file in the practice folder 'data folder' into a data frame, check if there is missing data, and decide how to replace it or delete it altogether with your colleague

**I** The results of the discussion should be organized through actual codes.

```
1 import pandas as pd
2 import seaborn as sns
```

```
1 a = pd.read_csv("../data/World University Rankings 2021/universities_ranking.csv")
```