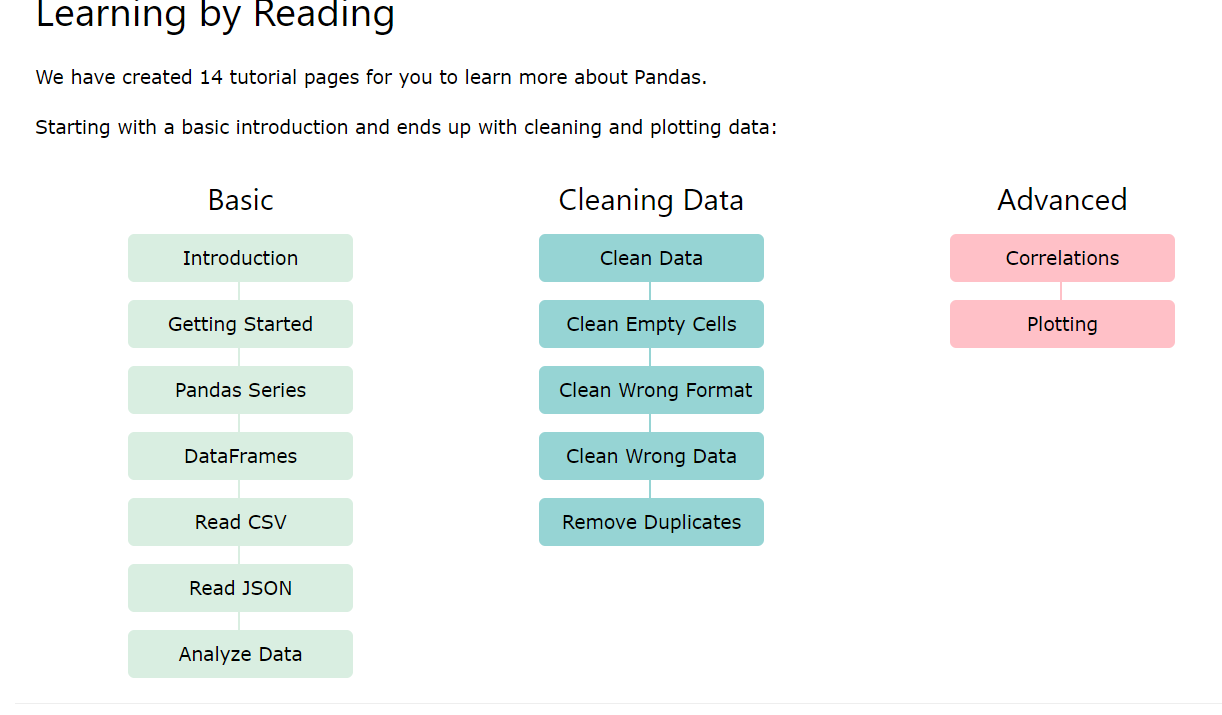
**Pandas in python …. class 1:**



<https://www.w3schools.com/python/pandas/default.asp>

## What is Pandas?

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

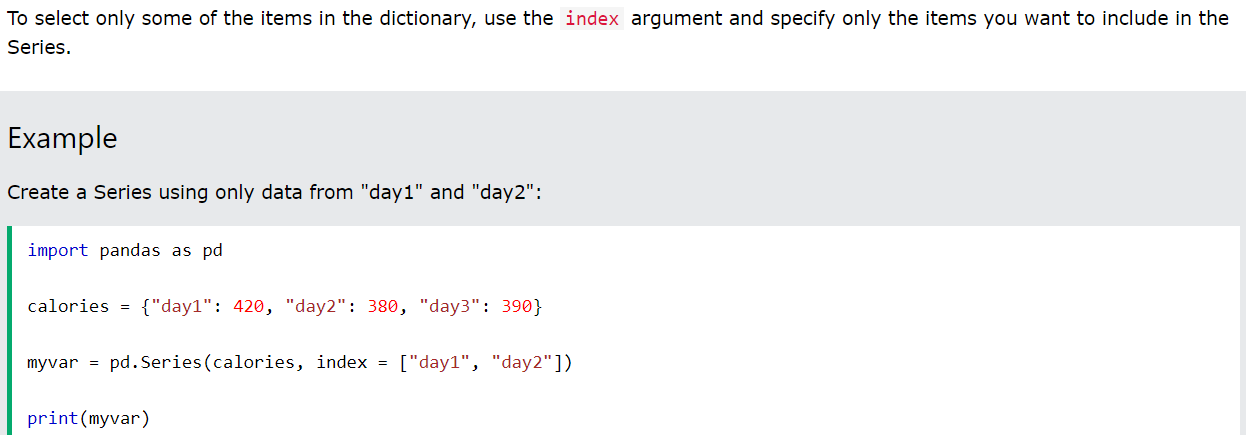
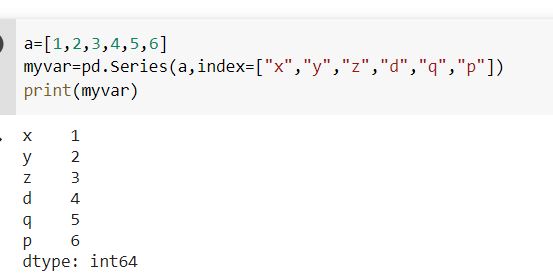
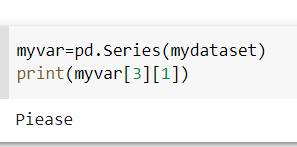
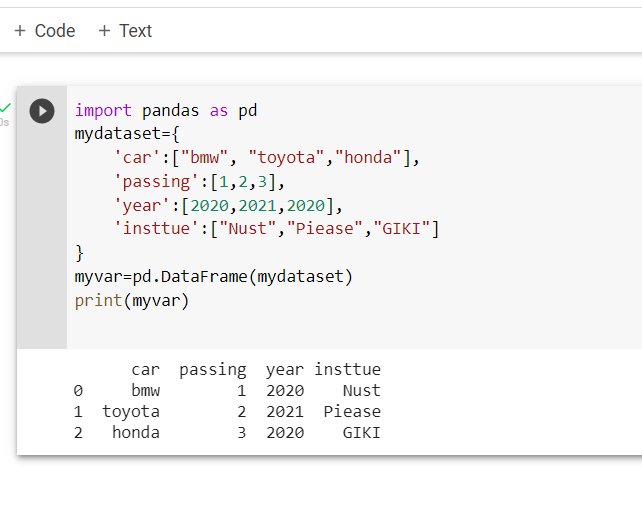
The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

## Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.



## DataFrames

Data sets in Pandas are usually multi-dimensional tables, called DataFrames.

Series is like a column, a DataFrame is the whole table.

### Example

Create a DataFrame from two Series:

import pandas as pd  
  
data = {  
  "calories": [420, 380, 390],  
  "duration": [50, 40, 45]  
}  
  
myvar = pd.DataFrame(data)  
  
print(myvar)

## What is a DataFrame?

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

### Example

Create a simple Pandas DataFrame:

import pandas as pd  
  
data = {  
  "calories": [420, 380, 390],  
  "duration": [50, 40, 45]  
}  
  
#load data into a DataFrame object:  
df = pd.DataFrame(data)  
  
print(df)



## Locate Row

As you can see from the result above, the DataFrame is like a table with rows and columns.

Pandas use the loc attribute to return one or more specified row(s)

### Example

Return row 0:

#refer to the row index:  
print(df.loc[0])

dataset={

    'Electronics':["jumperwire","arduino","esp8266"],

    'Prices':[450,550,700]

}

# to load the dataset into a fd variables

fd=pd.DataFrame(dataset)

print(fd.loc[[0,1]])

Electronics Prices 0 jumperwire 450

1 arduino 550

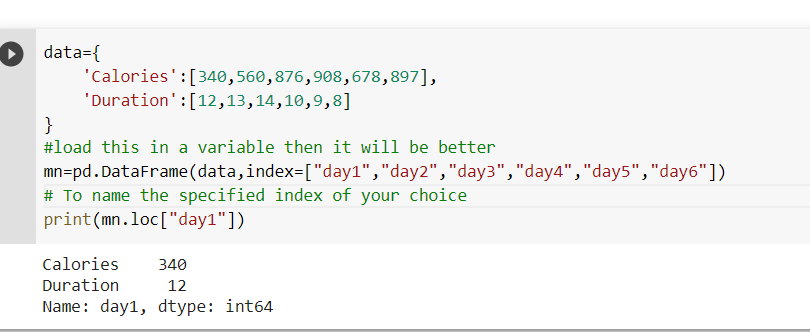
## Locate Named Indexes

Use the named index in the loc attribute to return the specified row(s).

### Example

Return "day2":

#refer to the named index:  
print(df.loc["day2"])



<https://www.w3schools.com/python/pandas/pandas_csv.asp>

## Dictionary as JSON

****JSON = Python Dictionary****

JSON objects have the same format as Python dictionaries.

If your JSON code is not in a file, but in a Python Dictionary, you can load it into a DataFrame directly:

### Example

Load a Python Dictionary into a DataFrame:

import pandas as pd  
  
data = {  
  "Duration":{  
    "0":60,  
    "1":60,  
    "2":60,  
    "3":45,  
    "4":45,  
    "5":60  
  },  
  "Pulse":{  
    "0":110,  
    "1":117,  
    "2":103,  
    "3":109,  
    "4":117,  
    "5":102  
  },  
  "Maxpulse":{  
    "0":130,  
    "1":145,  
    "2":135,  
    "3":175,  
    "4":148,  
    "5":127  
  },  
  "Calories":{  
    "0":409,  
    "1":479,  
    "2":340,  
    "3":282,  
    "4":406,  
    "5":300  
  }  
}  
  
df = pd.DataFrame(data)  
  
print(df)

data1={

    "Duration":{

        "0":60,

        "1":60,

        "2":45,

        "3":60,

        "4":60,

        "5":45

    },

    "pulses":{

        "0":110,

        "1":120,

        "2":115,

        "3":110,

        "4":120,

        "5":105

    },

    "MaxPulses":{

       "0":160,

        "1":120,

        "2":145,

        "3":160,

        "4":150,

        "5":145

    },

    "Calories":{

        "0":460,

        "1":560,

        "2":445,

        "3":760,

        "4":660,

        "5":545

    }

}

mj=pd.DataFrame(data1)

print(mj)

## Viewing the Data

One of the most used method for getting a quick overview of the DataFrame, is the head() method.

The head() method returns the headers and a specified number of rows, starting from the top.

### Example

Get a quick overview by printing the first 10 rows of the DataFrame:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
print(df.head(10))

There is also a tail() method for viewing the last rows of the DataFrame.

The tail() method returns the headers and a specified number of rows, starting from the bottom.

import pandas as pd

data1={

    "Duration":{

        "0":60,

        "1":60,

        "2":45,

        "3":60,

        "4":60,

        "5":45,

        "6":60,

        "7":45

    },

    "pulses":{

        "0":110,

        "1":120,

        "2":115,

        "3":110,

        "4":120,

        "5":105,

        "6":125,

        "7":145

    },

    "MaxPulses":{

       "0":160,

        "1":120,

        "2":145,

        "3":160,

        "4":150,

        "5":145,

        "6":160,

        "7":145

    },

    "Calories":{

        "0":460,

        "1":560,

        "2":445,

        "3":760,

        "4":660,

        "5":545,

        "6":760,

        "7":545

    }

}

mk=pd.DataFrame(data1)

print(mk)

## Data Cleaning

Data cleaning means fixing bad data in your data set.

Bad data could be:

* Empty cells
* Data in wrong format
* Wrong data
* Duplicates

## Remove Rows

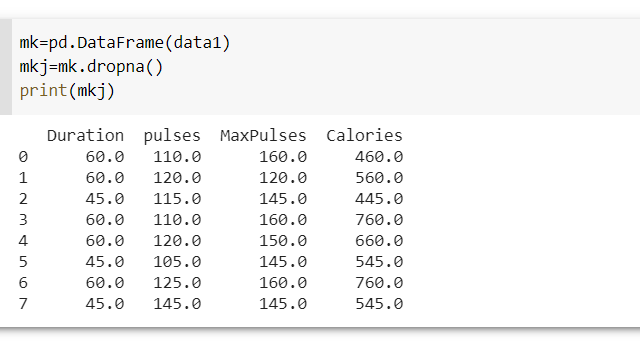
One way to deal with empty cells is to remove rows that contain empty cells.

This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

### Example

Return a new Data Frame with no empty cells:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
new\_df = df.dropna()  
  
print(new\_df.to\_string())



Empty cell

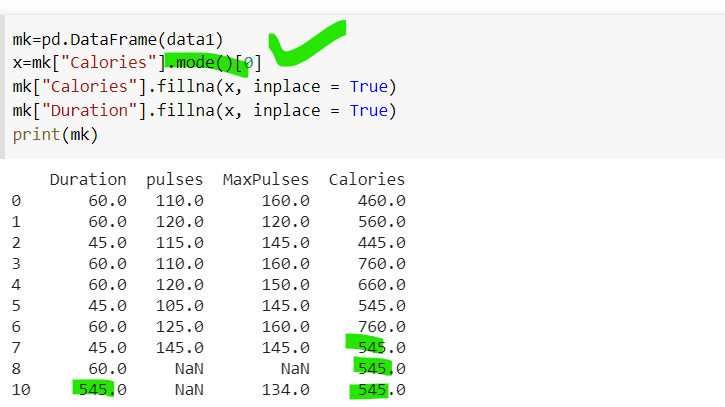
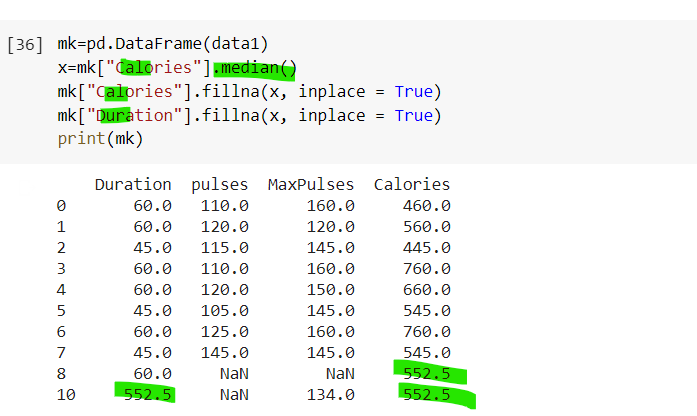
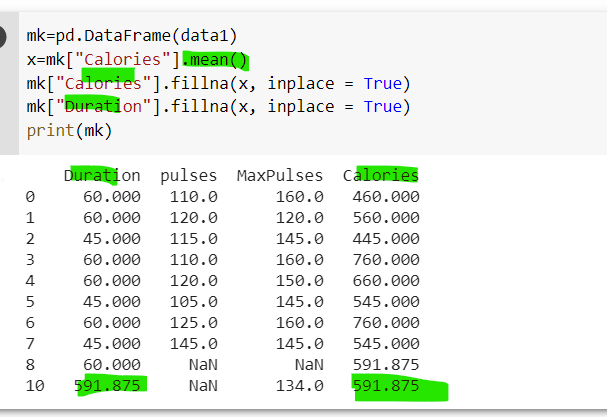
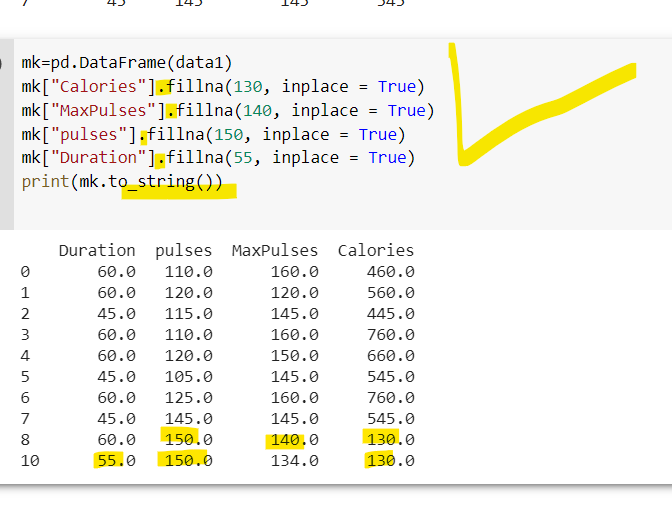
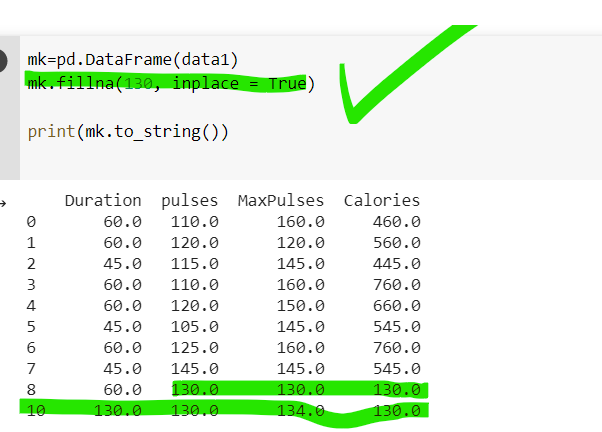
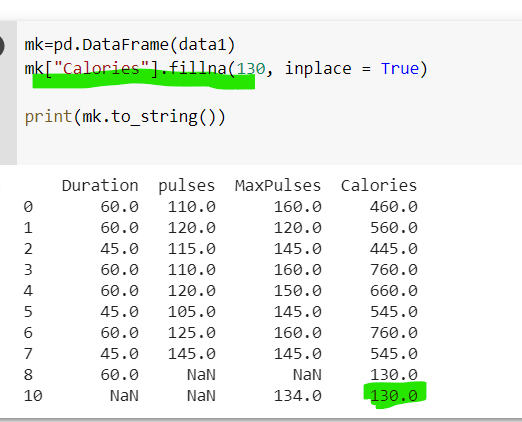
import pandas as pd

df = pd.read\_csv('data.csv')

df.fillna(170, inplace = True)

print(df.to\_string())

#Notice in the result: empty cells got the value 130 (in row 18, 22 and 28).



Let's try to convert all cells in the 'Date' column into dates.

Pandas has a to\_datetime() method for this:

### Example

Convert to date:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
df['Date'] = pd.to\_datetime(df['Date'])  
  
print(df.to\_string())

To discover duplicates, we can use the duplicated() method.

The duplicated() method returns a Boolean values for each row:

### Example

Returns True for every row that is a duplicate, othwerwise False:

print(df.duplicated())

## Removing Duplicates

To remove duplicates, use the drop\_duplicates() method.

### Example

Remove all duplicates:

df.drop\_duplicates(inplace = True)

## Finding Relationships

A great aspect of the Pandas module is the corr() method.

The corr() method calculates the relationship between each column in your data set.

The examples in this page uses a CSV file called: 'data.csv'.

## Plotting

Pandas uses the plot() method to create diagrams.

We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

Read more about Matplotlib in our [Matplotlib Tutorial](https://www.w3schools.com/python/matplotlib_intro.asp).

### Example

Import pyplot from Matplotlib and visualize our DataFrame:

import pandas as pd  
import matplotlib.pyplot as plt  
  
df = pd.read\_csv('data.csv')  
  
df.plot()  
  
plt.show()

## 

## Histogram

Use the kind argument to specify that you want a histogram:

kind = 'hist'

A histogram needs only one column.

A histogram shows us the frequency of each interval, e.g. how many workouts lasted between 50 and 60 minutes?

In the example below we will use the "Duration" column to create the histogram:

### Example

df["Duration"].plot(kind = 'hist')

