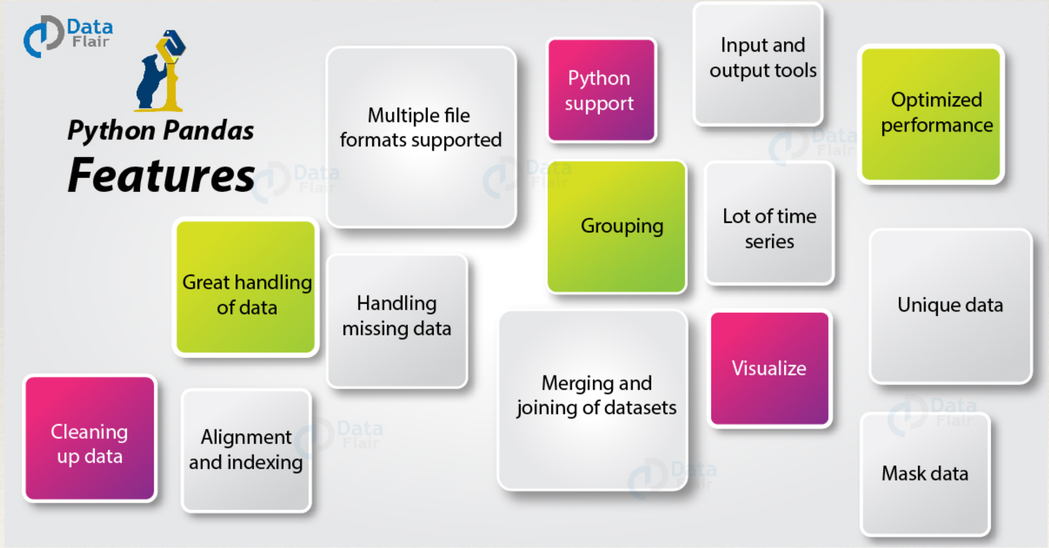
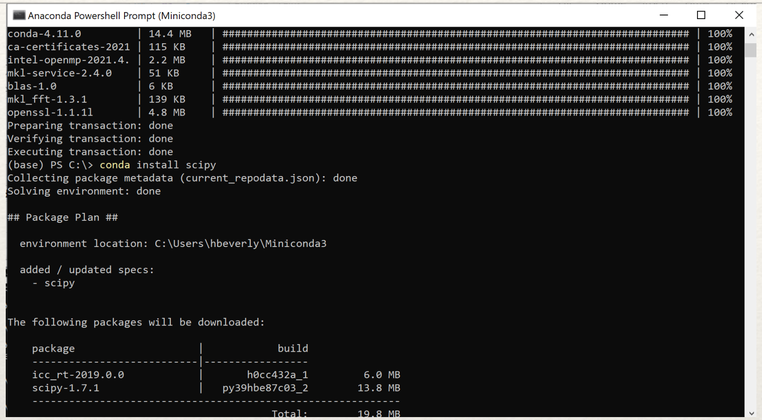
# What is pandas?

- A library for python
- Optimized for CPU usage so it can run on large data-sets
- Built on top of numpy and scipy
- Advanced data analysis
- Advanced features for data management
- Built-in stats / analysis functions



# Installing pandas
# + running from command shell




miniconda = a python environment

(start the miniconda powershell)

You can now run:

conda install pandas

**SUGGESTION: make a folder under C:\ called pythonprograms to store your programs AND data!**

cd pythonprograms

Now you can run your programs with

python programname.py

[important: you cannot use idle with miniconda…you can just type python from shell to get an interactive window]

# Let's Code!

## Use pandas to summarize stats about these two arrays.

arr1 = [1,3,5,6,7,19,23,55,777,34325,4346463]

arr2 = [4354,2342,645,34,4624,234,536,45,3,2,1]

Use a Series: https://www.datacamp.com/community/blog/python-pandas-cheat-sheet

# lambda functions

- A lambda function is a small anonymous function.
- A lambda function can take any number of arguments, but can only have one expression.
- ```
  My_function = lambda a : a + 10
  print(My_function(5))
  ```
  ```

# Pandas df.apply

- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.apply.html
- Apply a function along an axis of the DataFrame.
- Objects passed to the function are Series objects whose index is either the DataFrame's index (axis=0) or the DataFrame's columns (axis=1).

- df = pd.DataFrame([[4, 9]] * 3, columns=['A', 'B'])
- A B
- 0 4 9
- 1 4 9
- 2 4 9
- Using a reducing function from numpy (np)
- df.apply(np.sum, axis=0)
- A 12
- B 27
- df.apply(np.sum, axis=1)
- 0 13
- 1 13
- 2 13

# Computing and Adding new Columns

```
import pandas as pd

# make a simple dataframe

df = pd.DataFrame({'a':[1,2], 'b':[3,4]})

df

#    a  b

# 0  1  3

# 1  2  4
```

```
# create an unattached column with an index

df.apply(lambda row: row.a + row.b, axis=1)

# 0    4

# 1    6


# do same but attach it to the dataframe

df['c'] = df.apply(lambda row: row.a + row.b, axis=1)

df

#    a  b  c

# 0  1  3  4

# 1  2  4  6
```

# Let's Code!

## Use pandas to sort this table by (Price + Shipping Cost)

| Item | Price | Shipping |
|---|---|---|
| Car | 100000 | 50000 |
| Boat | 50000 | 125000 |
| Motorcycle | 25000 | 10000 |

Hints:

1. Use a DataFrame: https://www.datacamp.com/community/blog/python-pandas-cheat-sheet
2. Use apply and lambda function to make new column
3. Sort by new column

# Pandas is Excel? Yes! So why?

- A DataFrame is basically an excel sheet..
- In fact, Pandas can even more easily import csv and even import excel (.xls and .xlsx)
  - Csv directly into a DF: https://datatofish.com/import-csv-file-python-using-pandas/
  - Excel directly into a DF: https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html
    - *And export to cs: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html*
- So… what can you do with it?
  - Anything excel can do! Including column based operations, row based operations
    - *Column statistics, column/row slicing, formulaic manipulation and so much more*
    - **next week: visualization just like excel!*
- So why not use excel?
  - Excel does not have python!
  - Excel does not handle large datasets well [out of memory]
  - Excel does not take a file as an input parameter!
  - Excel doesn't connect directly to data sources and cannot stream analysis in realtime (pandas can!)

# Standard Deviation with Pandas

```
import pandas as pd

import numpy as np

#Create a DataFrame

d = {

    'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',

   'Rahul','David','Andrew','Ajay','Teresa'],

   'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],

   'Score2':[89,87,67,55,47,72,76,79,44,92,99,69],

   'Score3':[56,86,77,45,73,62,74,89,71,67,97,68]}

df = pd.DataFrame(d)

answer= df.std()

print("The standard deviations of the 3 columns are:")

print (answer)
```

# Correlation Coefficient

a number between −1 and +1 calculated so as to represent the linear dependence of two variables or sets of data.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

```
d = {  'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',
   'Rahul','David','Andrew','Ajay','Teresa'],
   'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],
   'Score2':[89,87,67,55,47,72,76,79,44,92,99,69],
   'Score3':[56,86,77,45,73,62,74,89,71,67,97,68]}
df = pd.DataFrame(d)
print(df.corr())
```

| | Score1 | Score2 | Score3 |
|---|---|---|---|
| Score1 | 1.000000 | 0.220204 | -0.097280 |
| Score2 | 0.220204 | 1.000000 | **0.390293** |
| Score3 | -0.097280 | 0.390293 | 1.000000 |