**Useful commands in Python**

1. Printing

print('Wage w: %5.2f.' % w\_ss)

print(f’Wage w: {w}’)

print(f'Representing: {wtd\_tot[year] \* 1e-5:,.2f} Lakh taxpayers')

1. Numpy one dimensional matrix

nvec = np.array([1.0, 1.0, 0.2])

two dimensional matrix

bmat = np.zeros((10, 20))

1. Numpy All rows except last

bmat[:-1, :]

1. Numpy last element

A[-1]

Second last element A[-2]

1. Numpy, rowwise summing across columns - hotizontally

np.sum(bmat, axis=1)

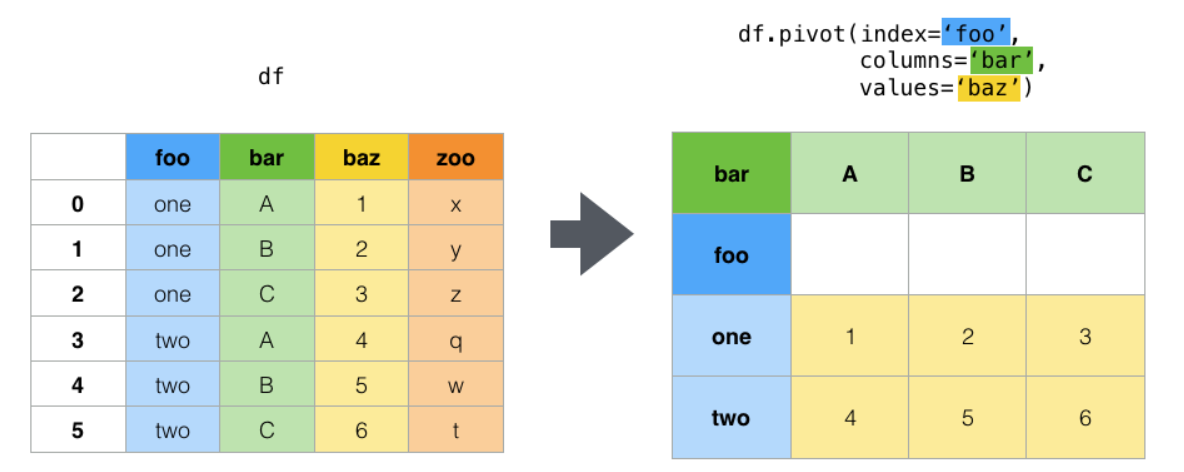
bmat.sum(axis=1)

summing across rows, axis = 0

1. Reshape

etr\_vec = etr\_vec.reshape(etr\_vec.shape[0], 1)

1. Reshape a dataset - pivot



1. Matrix multiplication

A = np.array([[1, 2, 3]])

B = np.array([[4],[5],[6]])

B.dot(A) =

array([[ 4, 8, 12],

[ 5, 10, 15],

[ 6, 12, 18]])

1. Adding a new column to a dataframe

# Define a dictionary containing Students data

data = {'Name': ['Jai', 'Princi', 'Gaurav', 'Anuj'],

        'Height': [5.1, 6.2, 5.1, 5.2],

        'Qualification': ['Msc', 'MA', 'Msc', 'Msc']}

# Convert the dictionary into DataFrame

df = pd.DataFrame(data)

# Declare a list that is to be converted into a column

address = ['Delhi', 'Bangalore', 'Chennai', 'Patna']

# Using 'Address' as the column name

# and equating it to the list

df['Address'] = address

1. Adding a new column to a dataframe and filling it with the same value:

D = {'newcol1': 'a',

'newcol2': 2,

'newcol3': 1}

df = pd.DataFrame({'A':[1,2],

'B':[4,5],

'C':[7,8]})

print (df)

A B C

0 1 4 7

1 2 5 8

print (pd.concat([df, pd.DataFrame(D, index=df.index)], axis=1))

A B C newcol1 newcol2 newcol3

0 1 4 7 a 2 1

1 2 5 8 a 2 1

Or use

A.assign(\*\*d)

A = pd.DataFrame(np.random.rand(10, 5), columns=list('abcde'))

d = {

'newcol1': 'a',

'newcol2': 2,

'newcol3': 1

}

A.assign(\*\*d)

a b c d e newcol1 newcol2 newcol3

0 0.709249 0.275538 0.135320 0.939448 0.549480 a 2 1

1 0.396744 0.513155 0.063207 0.198566 0.487991 a 2 1

2 0.230201 0.787672 0.520359 0.165768 0.616619 a 2 1

3 0.300799 0.554233 0.838353 0.637597 0.031772 a 2 1

4 0.003613 0.387557 0.913648 0.997261 0.862380 a 2 1

5 0.504135 0.847019 0.645900 0.312022 0.715668 a 2 1

6 0.857009 0.313477 0.030833 0.952409 0.875613 a 2 1

7 0.488076 0.732990 0.648718 0.389069 0.301857 a 2 1

8 0.187888 0.177057 0.813054 0.700724 0.653442 a 2 1

1. 0.003675 0.082438 0.706903 0.386046 0.973804 a 2 1
2. Type of columns of a dataframe

df.dtypes

1. df.describe()
2. Drop rows

df[df.name != 'Tina']

1. Dropping columns

df.drop('column\_name', axis=1, inplace=True)

To drop by column number instead of by column label, try this to delete, e.g. the 1st, 2nd and 4th columns:

df = df.drop(df.columns[[0, 1, 3]], axis=1)

1. Drop missing rows (NaN) and plot

df = df.dropna(subset=['ShiftedPrice'])

df.plot(x='date', y='ShiftedPrice')

1. Useful Print Command to print both text and formatted numbers

Year and total\_consumption\_all1 are variables

print(f'Total Consumption in - {year}: {total\_consumption\_all1:,.0f}')

1. Concatenate and Appending Dictionaries and creating json files

# dict1 = df1.to\_dict()

# dict2 = df2.to\_dict()

# Concatenate two dictionaries

# dict1.update(dict2)

# dictf = {"read": dict1, "calc": dict2}

#converting the dictionary to dataframe

# df\_gst\_for\_json = pd.DataFrame(dict\_gst\_rec)

# df = pd.DataFrame(dictf)

# saving dataframe to json file

# df\_gst\_for\_json.to\_json("gst\_rec.json")

# df.to\_json("gst\_records.json")

# Merging the two dictionary along with adding "read" and "calc"

dict\_gst\_rec = {"read": dict\_gst\_read, "calc": dict\_gst\_calc}

# Pretty Print dictionary into json file

with open("gstrecords\_variables.json", "w") as f:

json.dump(dict\_gst\_rec, f, indent=4, sort\_keys=True)

1. Sort by a column

df.sort\_values(by=['col1'])

1. Selecting rows based on multiple criteria

revenue\_df[(revenue\_df['Region\_Code']=="LAC") & (revenue\_df['year']==2016)][['Country', 'GDP\_PC']]

1. Changing name of a column programmatically

df.columns = df.columns.str.replace('CONS\_','GST\_')

1. Adding up columns programmatically

In this example, we add up all the columns with name starting with ‘CONS\_’ and allocates it to ‘CONS\_OTHER’

df['CONS\_OTHER'] = df[df.columns[df.columns.str.startswith('CONS\_')]].sum(axis=1)

1. Creating blank rows in a dataframe

df = pd.concat([df, pd.DataFrame([[np.nan] \* df.shape[1]], columns=df.columns)], ignore\_index=True)

1. Histograms

import seaborn as sns

df\_survey\_mpc = df\_cons\_summ\_all[df\_cons\_summ\_all["Srl\_no"] == 49]

df\_survey\_mpc.index = pd.RangeIndex(len(df\_survey\_mpc.index))

df\_survey\_mpc = df\_survey\_mpc.drop(df\_survey\_mpc.columns[0], axis = 1)

sns.distplot(tuple(df\_survey\_mpc['Value']), hist = False, kde = True, kde\_kws = {'linewidth': 3}, label = "consumption")

1. Extract certain Columns (and otherwise) from a dataframe

df1 = pd.DataFrame(np.array([[1, 0, 3], [4, 5, 6]]))

df1.columns=['A', 'B', 'C']

change\_cols = ['A', 'C']

df1 = df1[change\_cols]

This will give a ‘SettingWithCopyWarning’, so use deepcopy instead

df1 = df[cols].copy(deep=True)

for all columns except ['A', 'C']

df1[df1.columns.difference(change\_cols)]

Again, this will give a ‘SettingWithCopyWarning’, so use deepcopy instead

df2 = df[df1.columns.difference(change\_cols)].copy(deep=True)

1. Join one dataframe to another (concat)

df = pd.concat([df2, df1], axis=1)

1. Inserting a character in a string

S is a string, this command inserts a ‘,’ after the third character of S

S = S[:3] + ',' + S[3:]

1. Convert a dataframe row to a list,

Here row ‘i’ of the dataframe df is converted into a list and assigned to ‘row’

row = df.loc[i].values.tolist()

1. Creating lists, appending and inserting elements

row = []

row.append('A')

['A']

say row=['A', 'B', 'C']

row.insert(1,’A1’)

['A', 'A1', 'B', 'C']

1. Inserting a list into a dataframe at a particular row position (replacing the row)

This command inserts a list ‘row’ at row ‘i’ of the dataframe df

df.loc[i] = row

1. Merging dataframes

df\_rates = pd.merge(df\_rates, df\_product, how="inner", on="product\_id")

1. Rename dataframe columns

pit\_df = pit\_df.rename(columns={'GTI':'GTI\_1', 'pitax':'pitax\_1'})

1. Transpose a dataframe

df = df.T

1. Dictionary to dataframe and then transpose

inc\_block\_df = pd.DataFrame.from\_dict(inc\_block, orient='columns').T

1. Replace one column with a manipulation of another column

inc\_block\_df['year']=inc\_block\_df['year'].str[:4]

strip the leading and following spaces

fdi\_df['Country'] = fdi\_df['Country'].str.strip()

1. Find Replace with condition – Complex

fdi\_df['Region\_Code\_1'] = np.where(fdi\_df['Country\_Code']=="BRA", "BRA", fdi\_df['Region\_Code'])

1. Find Replace with condition – Simple

fdi\_df['Region\_Code\_1'].replace(['EAP'],'BRA')

1. Filter or Select rows of a dataframe

dumpdf\_1[['CIT\_ID\_NO', 'sector', 'citax']][dumpdf\_1['citax']>0]

1. Remember in lists

row=['A', 'B', 'C']

row[0] = ‘A’ , a string

while row[0:1] = [‘A’] a list

1. IMPORTANT

Copying a list is not straightforward

list\_tdict\_orig = list\_tdict[:] (list\_tdict\_orig = list\_tdict is NOT CORRECT)

1. Remove an item from a list

row=['A', 'B', 'C']

row.pop(1)

'B'

row

['A', 'C']

1. Select a random number of rows from a dataframe

df1 = df.sample(n=2000)

1. Convert the index into a field

df[‘index1’] = df.index

also

df.reset\_index(level=0, inplace=True)

1. Working with Dictionaries

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

thisdict.items

* dict\_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])

thisdict.keys()

* dict\_keys(['brand', 'model', 'year'])

thisdict.values()

* dict\_values(['Ford', 'Mustang', 1964])

thisdict["model"] = ‘Mustang’

Change Value

* thisdict["year"] = 2018

for x in thisdict.values():  
  print(x)

'Ford'

'Mustang'

1964]

Looping through keys and values

for x, y in thisdict.items():  
  print(x, y)

To determine if a specified key is present in a dictionary use the

if "model" in thisdict:  
  print("Yes, 'model' is one of the keys in the thisdict dictionary")

length of a dictionary

print(len(thisdict))

Adding an item to the dictionary is done by using a new index key

thisdict["color"] = "red"  
print(thisdict)

Removing items

thisdict.pop("model")  
print(thisdict)

Removes the last inserted item

thisdict.popitem()  
print(thisdict)

Emptying a dictionary

thisdict.clear()

Constructing a dictionary

thisdict = dict(brand="Ford", model="Mustang", year=1964)

Makes a dictionary from a list

dict(list\_tdict)

Make a dictionary into a list

list(tdict)

1. Filling in a dictionary (example)

wtd\_tax\_clp={}

wtd\_tax\_clp[year] = calc1.weighted\_total('pitax')

Access the elements using wtd\_tax\_clp[year]

1. Transposing a dictionary

from collections import defaultdict

ref=defaultdict(dict)

for j in range(ref\_num):

i=0

for pkey, sdict in a.items():

print(f'pkey: {pkey}')

print(f'sdict: {sdict}')

print(sdict[j])

ref[j][year+i] = sdict[j] (in this example we transpose the dictionary)

i=i+1

1. Accessing a dictionary

d = dict((k, v) for k, v in reform[i].items() if k >= BASE\_YEAR)

In this example we only select items that are greater than the BASE\_YEAR

1. Summing groups in pandas

You can select the columns of a groupby:

In [11]: df.groupby(['Country', 'Item\_Code'])[["Y1961", "Y1962", "Y1963"]].sum()

Out[11]:

Y1961 Y1962 Y1963

Country Item\_Code

Afghanistan 15 10 20 30

25 10 20 30

Angola 15 30 40 50

25 30 40 50

Also,

df.groupby(['Country', 'Item\_Code']).agg({'Y1961': np.sum, 'Y1962': [np.sum, np.mean]}) # Added example for two output columns from a single input column

The category grouped by would be pushed into the index. You would need to do a df.reset\_index() to make it a column.

1. Get Attributes of a Class or any Object

print(dir(a))

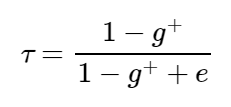
print(vars(a))

**HMTL**

1. Printing special characters from Word imported into html

To center use \[

\[ τ = \frac{1 - g^+}{1 - g^+ +e}\]

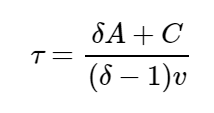


This is aligned with the text – we use \(

\( τ = \frac{1 - g^+}{1 - g^+ +e}\)

\(a^2, e^x\)

\[τ = \frac{δA+C}{(δ-1) v}\] in text gives (centered)



in the webpage.

1. **File Writing and Reading**

# Writing to a File

f= open("guru99.txt","w+")

f.write("This is line %d\r\n" % (i+1))

#Appending to a file

f=open("guru99.txt", "a+")

f.close()

# Reading a File

With open ("microsimulation\_instructions.txt"), ‘r’) as f:

contents = f.read()

print(contents)

#line by line

with open('test.txt','r') as f:

lines = f.readlines()

for line in lines:

print(line.rstrip('\n'))

filepath = 'Iliad.txt’

with open(filepath) as fp:

line = fp.readline()

cnt = 1

while line:

print("Line {}: {}".format(cnt, line.strip()))

line = fp.readline()

cnt += 1

1. **Regression**

X = df["RM"] ## X usually means our input variables (or independent variables)

y = target["MEDV"] ## Y usually means our output/dependent variable

X = sm.add\_constant(X) ## let's add an intercept (beta\_0) to our model

model = sm.OLS(y, X).fit()

predictions = model.predict(X) # make the predictions by the model