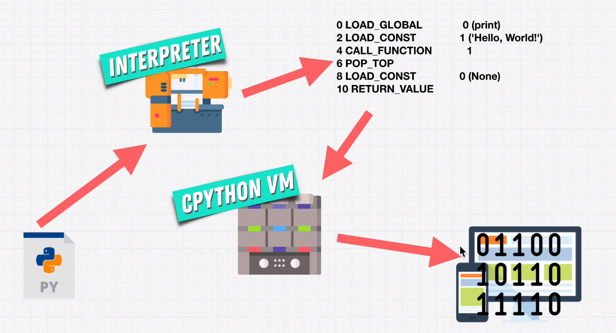
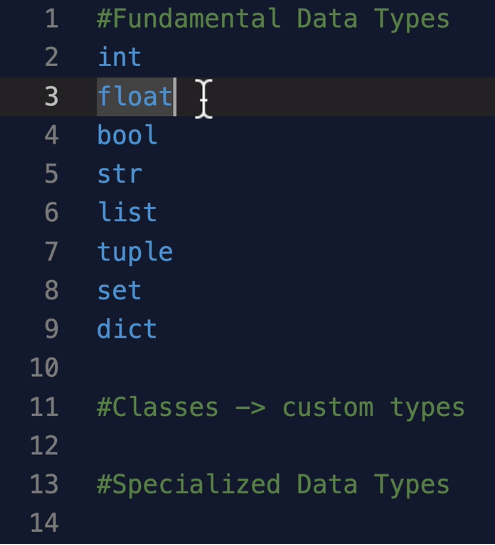
1. What is Python framework?

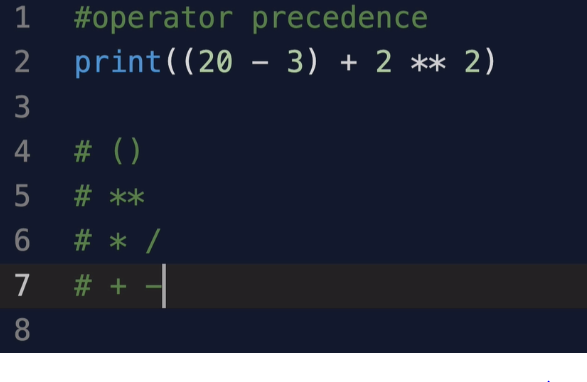


1. Python DataTypes:

Below are Python Datatypes



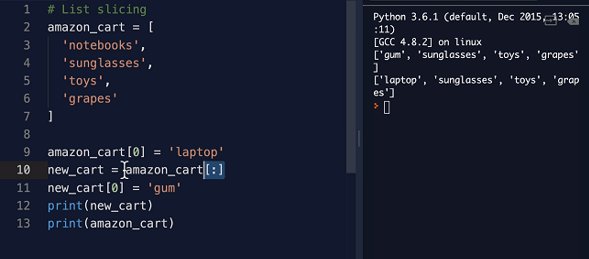
Operator precedence in Math:



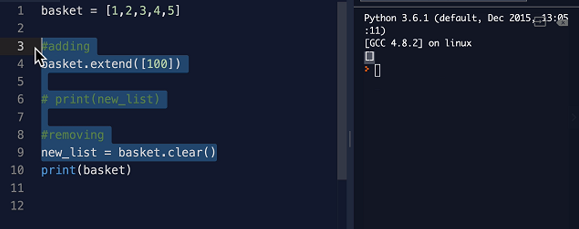
How to reverse and play out with the order of strings



List Slicing:



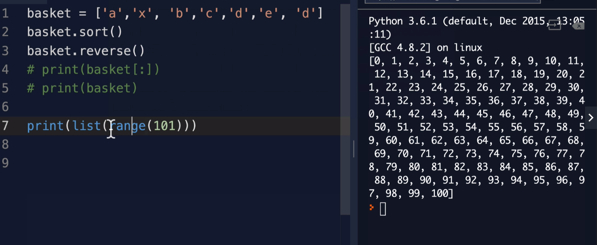
List Methods:



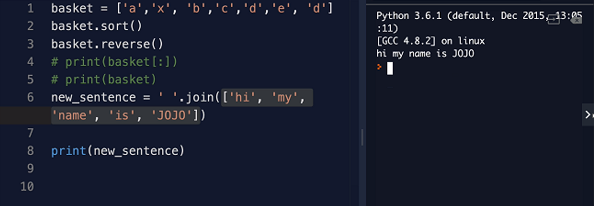
Sorted Reverse order:



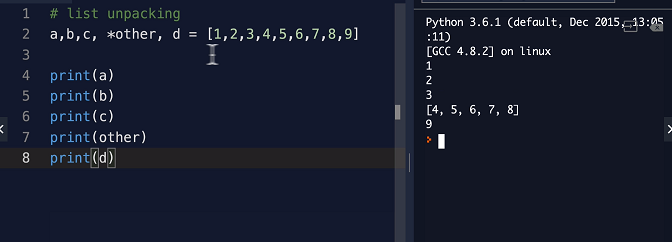
PRINT RANGE:



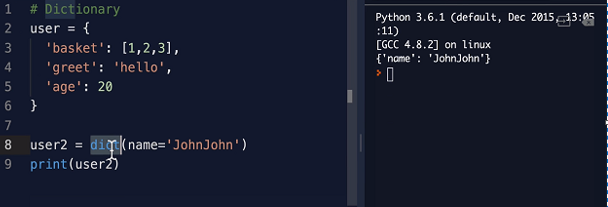
LIST JOIN:



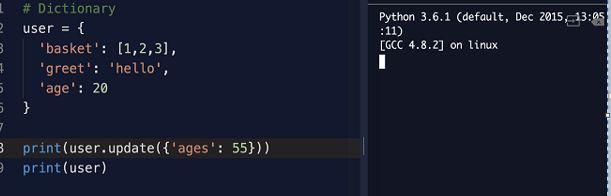
LIST Unpacking:



Dictionary methods:

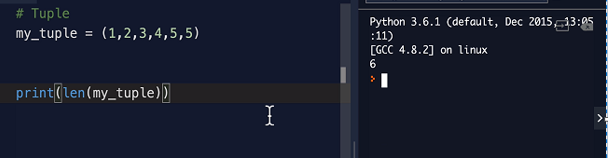


Dictionary methods update:



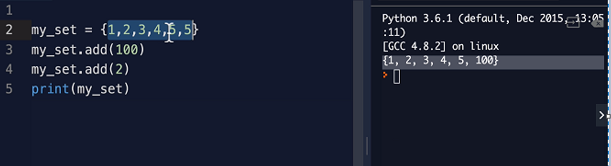
Tuple:

Tuple are mutable whereas List are immutable:



SETS:

Set has unordered unique set of records and cannot have duplicates



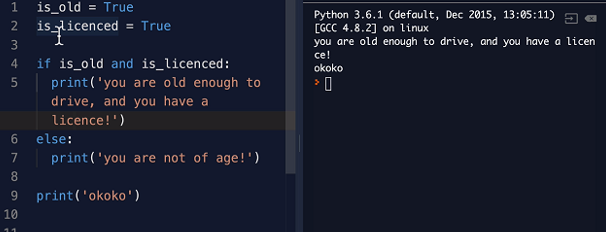
To get a unique set of records from a list



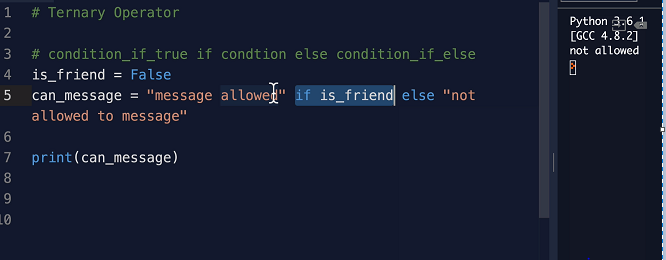
Set methods:



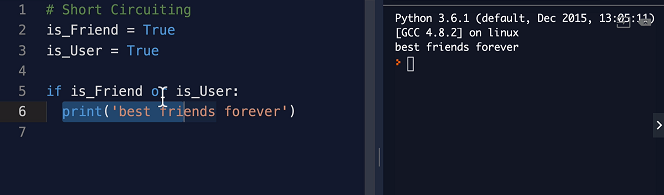
IF statement:



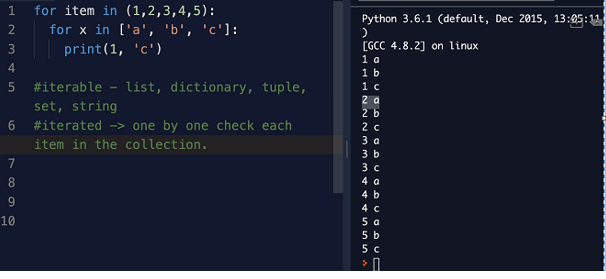
Ternary Operator for If Condition:

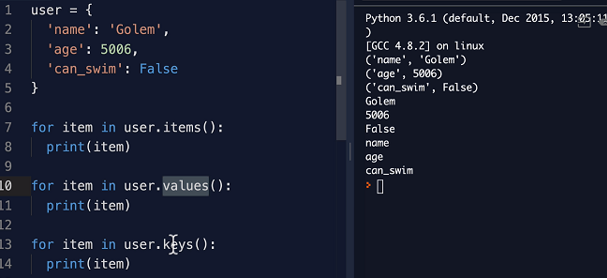


Short Circuiting ( OR statement):

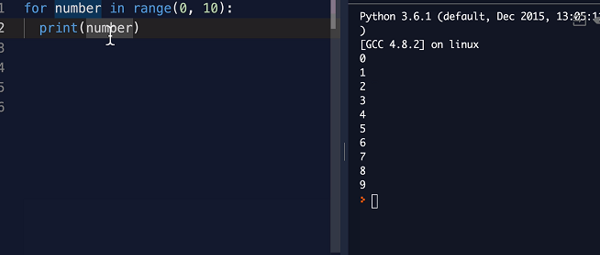


FOR loop and iterables:

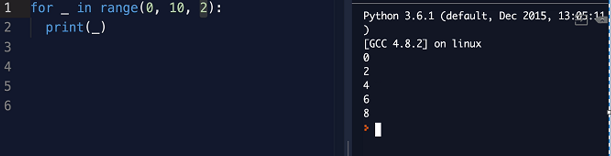




For number in range:

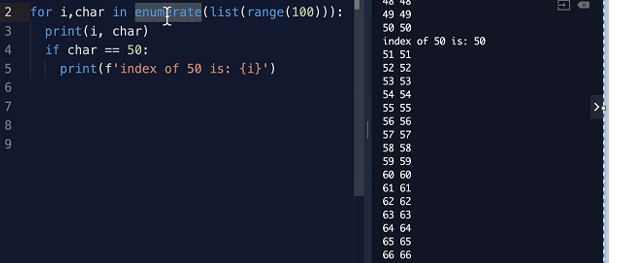


Range with stepover option

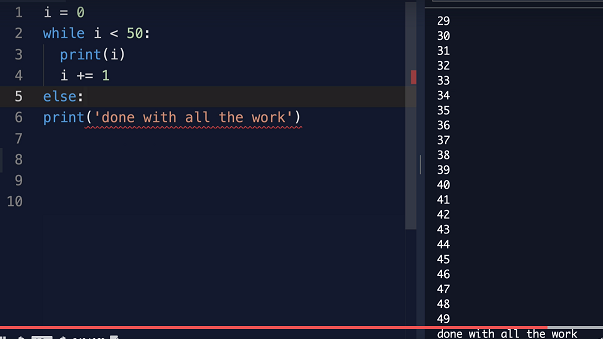




Enumerate:

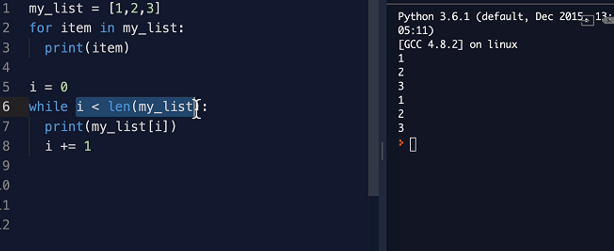


While Loop:

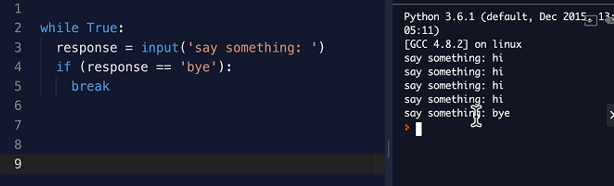


For Loop Vs While Loop:

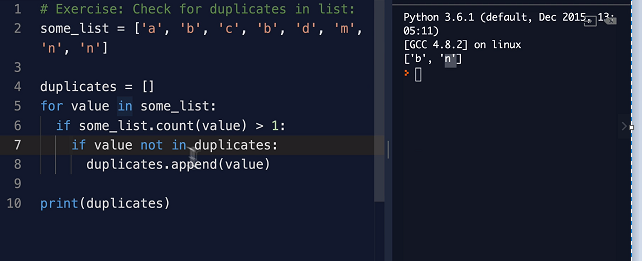
Use For for finite list, While for infinite or big list.



Use while loop for example in “enter correct password” for a user:

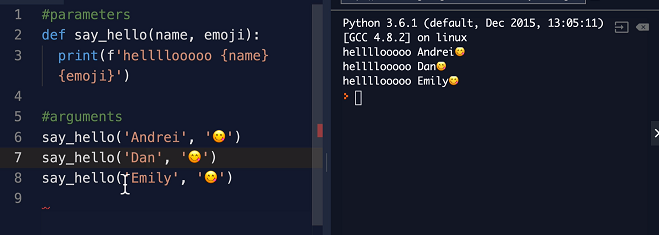


**Find duplicates in a LIST:**

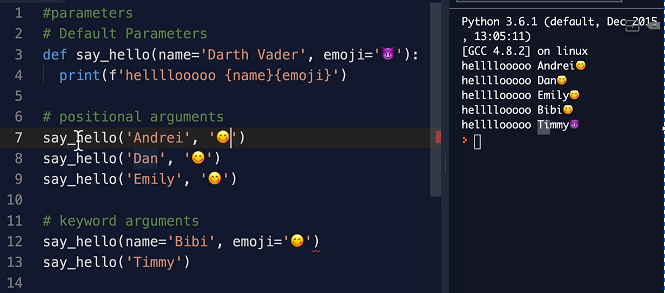


**Functions with parameters and arguments:**

**Parameters are defined in the function and arguments are passed to the function.**



**Functions with default and parameters:**



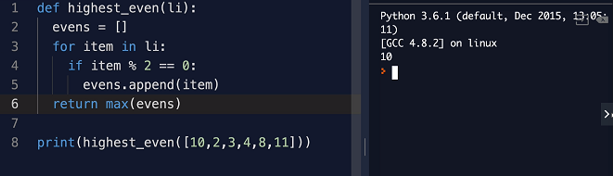
**To check if number is even or odd**



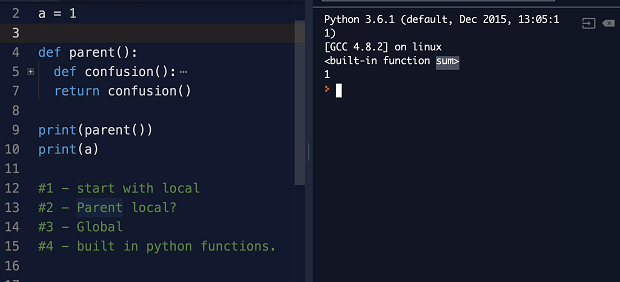
**Args, Keyword Args kwargs**



**Exercise :** Find the highest even from a list:



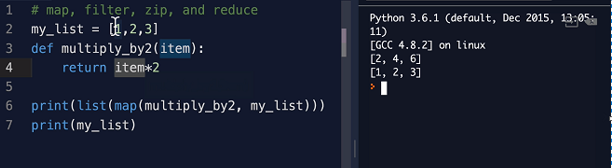
**SCOPE Rules:**



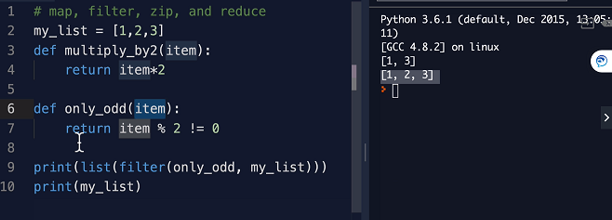
**]**

**MAP function:**

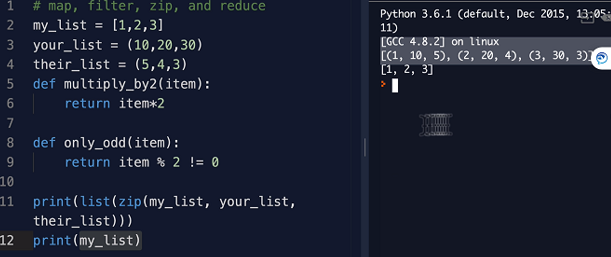
**It will do the mapping functionality.**



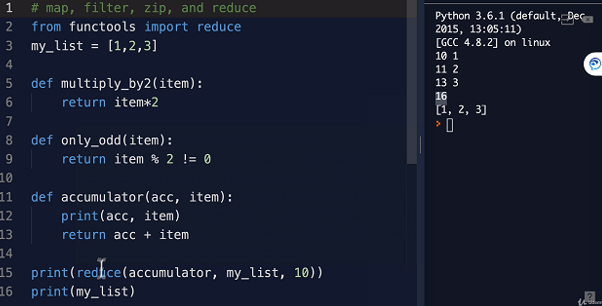
Filter() function:



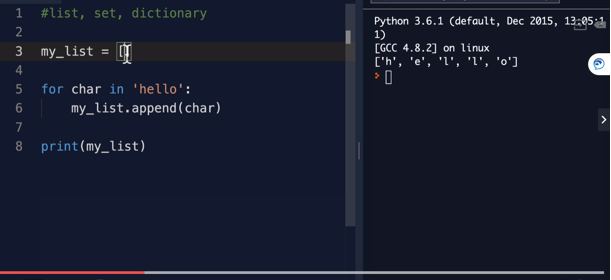
**Zip() Function:**



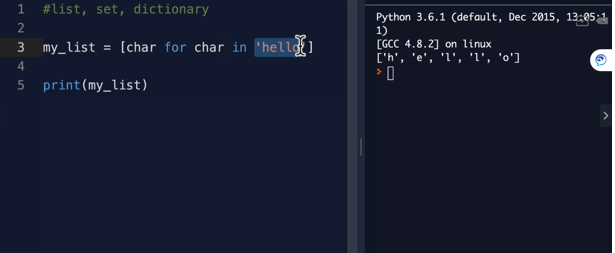
**Reduce() function:**



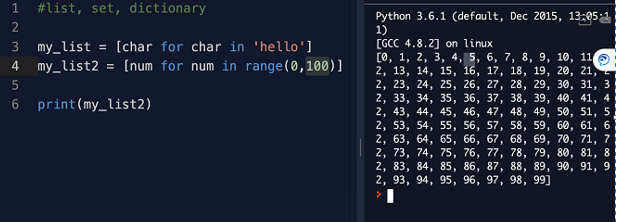
**LIST COMPREHANRSION:**



**Above can be rewritten with list comprehension:**



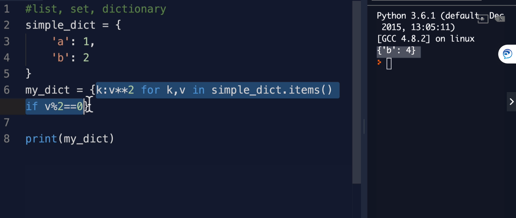
**Number with range**

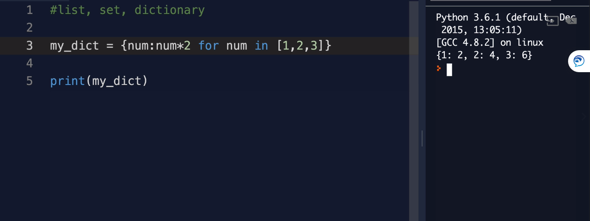


**Different math concepts with list comprehension**



For Dictionary:



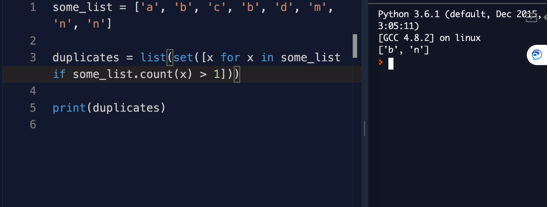


Exercise:

Convert below for finding duplicates into comprehension:



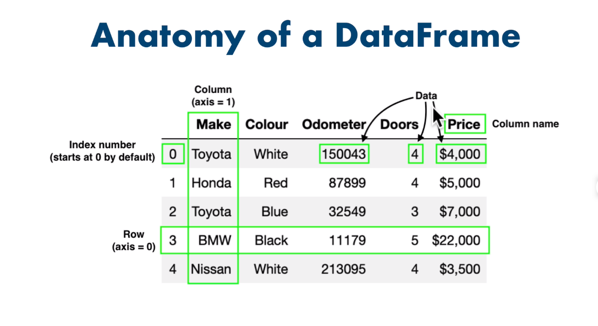
**Below with comprehension:**



**Pandas – Concepts and commands**

1. **How to import csv file into pandas:**

Car\_sales = pd.read\_csv(“car-sales.csv”) (use tab to import the file)



**Export to csv file**

Car\_sales = pd.to\_csv(“car-sales\_new.csv”,index= False)

**Pandas functions to explore data:**

Car\_Sales.describe()

Car\_sales.info()

Car\_Sales.mean()

Car\_sales[“Door”].sum() 🡪 This will give sum of a particular column

Len(car\_sales)

Car\_sales.head(7) 🡪 This will give top 7 rows

Car\_sales.tail(7) 🡪 This will give bottom 7 rows

Animals = pd.Series(“cat”,”dog”,’”snake”,index=[0,2,5])

Animals.loc[3] 🡪 This will pull the data in that index location of 3.

Animals.iloc[3] 🡪 THi swill give postion data rather than index position

**List Slicing:**

Car\_sales.iloc[:3] 🡪 This will give upto index 3

**Car\_sales.Make and car\_sales[“Make”]** will give same results

**Filters:**

Car\_sales[car\_sales[“Make”] == “Toyota”]

Car\_sales[car\_sales[“odometer (km)”] > 10000 ]

**For Matrix view:**

For 2 columns comparison

Pd.crosstab(car\_sales[“Make”] , car\_sales[“Doors”])

Above will give a matrix view of the data.

**For Group by:**

Car\_sales.groupby([“Make”]).mean()

Can we do some visualization:

Car\_sales[“odometer (KM)”].plot()

If the plot doesn’t show up , use below before doing the plot

%matplotlib inline

Import matplotlib.pyplot as plt

To show histogram:

Car\_sales[“Odometer (KM)”].hist()

**To convert dollar amount to integer:**

Car\_Sales[“price”] = car\_sales[“price”].str.replace(‘[\$\,\.], ‘’).astype(int)

**Manipulating data:**

Car\_sales[“Make”].str.lower()

**It will not save so we need to reassign**

Car\_sales[“Make”] **=** Car\_sales[“Make”].str.lower()

And then again execute Car\_sales

**Now how to populate null values in pandas dataframe with a mean value (null values in pandas is putup as ‘NaN’)**

Car\_Sales\_missing[“odometer”].fillna(car\_sales\_missing[“odometer”].mean(),inplace = True)

**Now to how to drop rows with null values:**

Car\_sales\_missing\_dropped = car\_sales\_missing.dropna()

Now write the data with dropped null values to a new file:

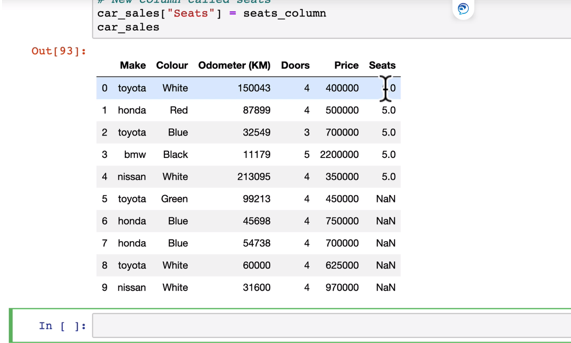
Car\_sales\_missing\_dropped.to\_csv(“car\_sales\_missing\_droppped.csv”)

**Create a new column called seats**

Seats\_column = pd.Series([5,5,5,5,5])

Car\_sales[“seats”] = Seats\_column

Car\_sales



**Add column by column from list**

**#column from python list**

Fuel\_economy = [7,1,2,3,5,3,4,6,4]

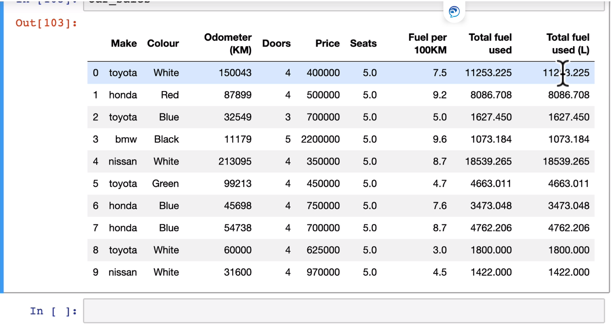
Car\_sales[“fuel per 100KM”] = fuel\_economy

Car\_sales

**Now to create another column using operations:**

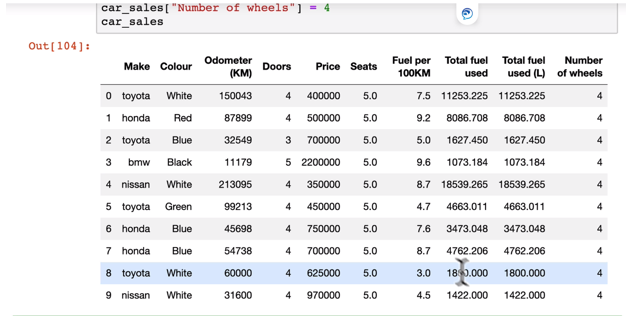
Car\_sales[“Total fuel used”] = car\_sales[“fuel per 100KM” ]/100 \* car\_sales[“fuel per100KM”]

Car\_sales



**Another way of creating a column**

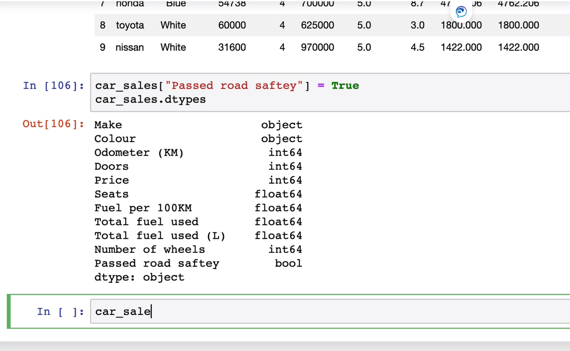
Car\_Sales[“Number of seats”] = 4



**Adding column for by Boolean value**

Car\_Sales[“Passed Road Safety”] = True

Car\_Sales.dtypes



**To Drop a column:**

Car\_sales.drop(“Total fuel used” , axis = 1 )

**Now how to shuffle the incoming data in order to train the machine learning model as the order of data will spoil the data set, we need to shuffle the data in order to maintain randomness else the traning set data will be mostly predictable.**

Car\_sales.sample(frac=1)

Car\_sales\_shuffled = Car\_sales.sample(frac=1)

**Car\_sales\_shuffled**

🡪 frac= 1 means shuffle 100% of the data , if it is 0.5 then it will shuffle 50% of the data.

If fata is large then we can shuffle on 10% of the data.

Car\_sales.sample(frac=0.01)

**To reset the index position**

Car\_sales\_shuffled.reset\_index(drop = True,inplace = True)

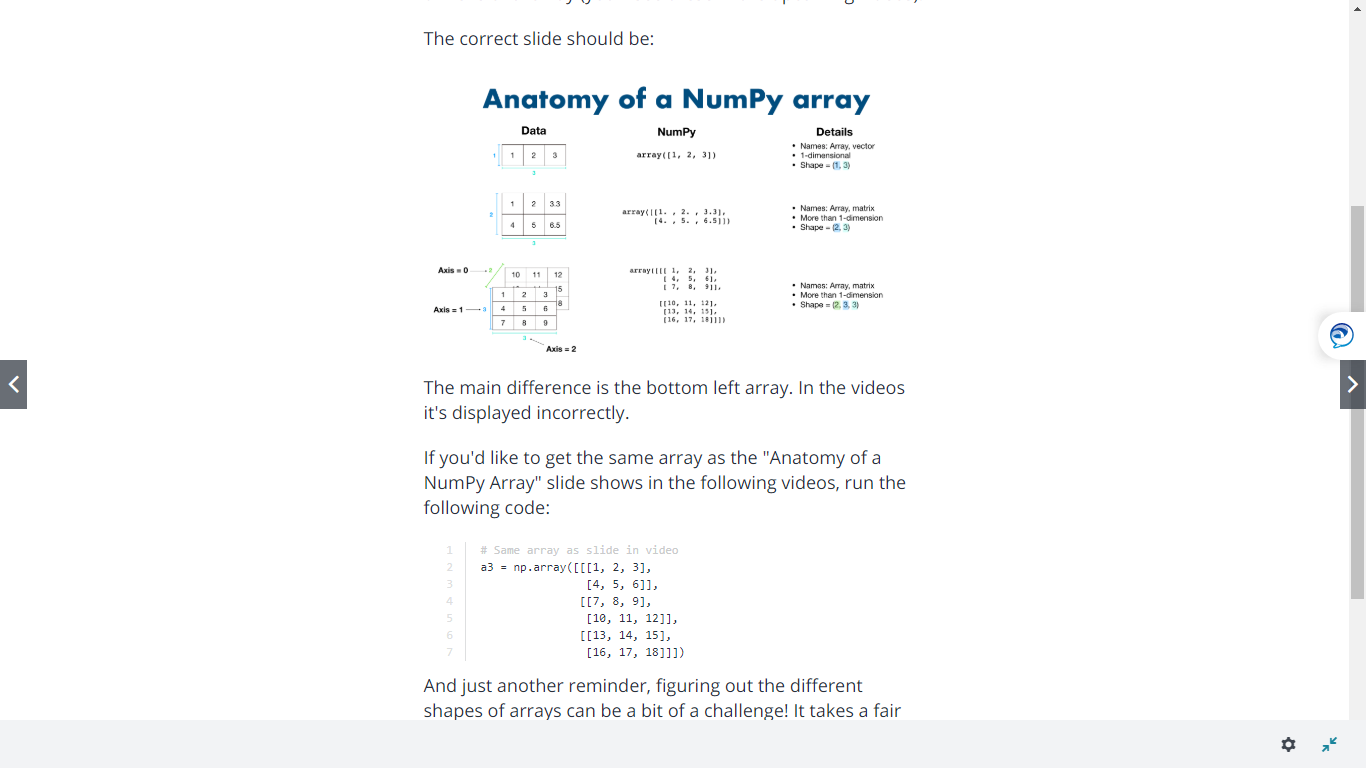
**How to apply operation through lambda function:**

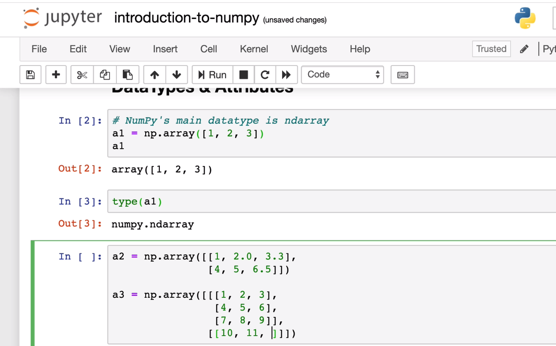
Car\_sales[“odometer (KM)”] = Car\_sales[“odometer (KM)”].apply(lambda x:x/1.6)

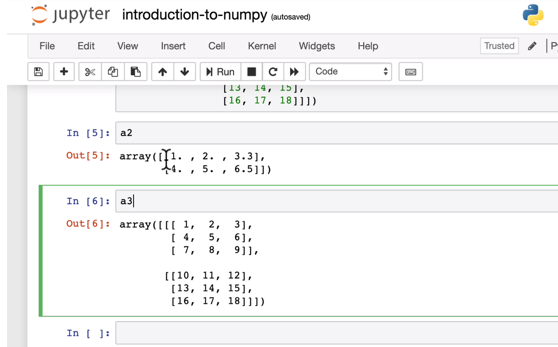
Car\_Sales

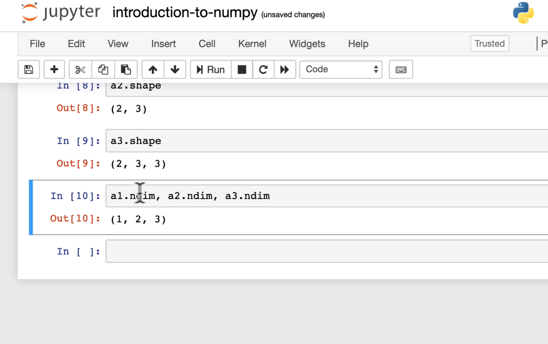
**Above will change kms to miles.**

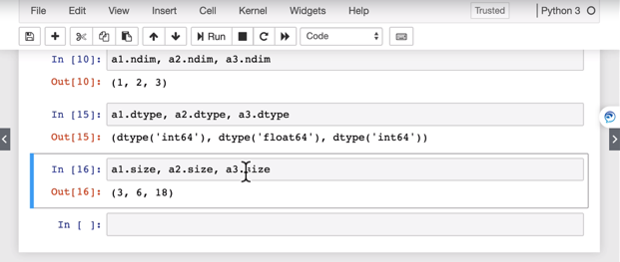
**Numpy Concepts:**

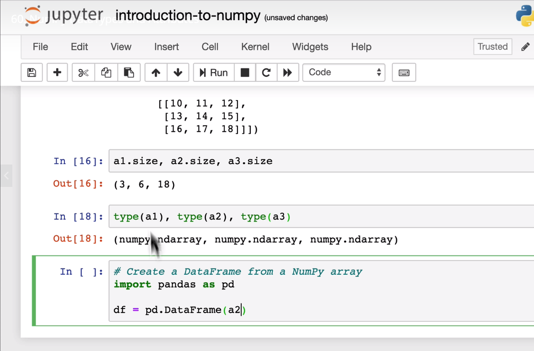


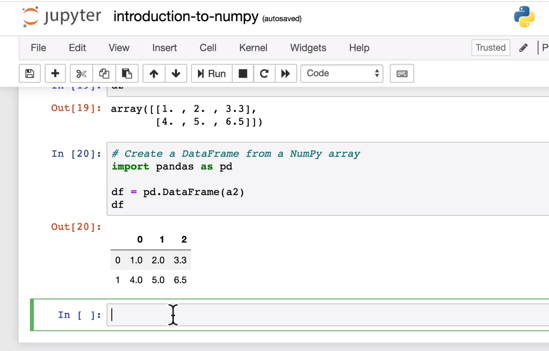












**Creating Arrays:**

Ones = np.ones((2,3))

Ones

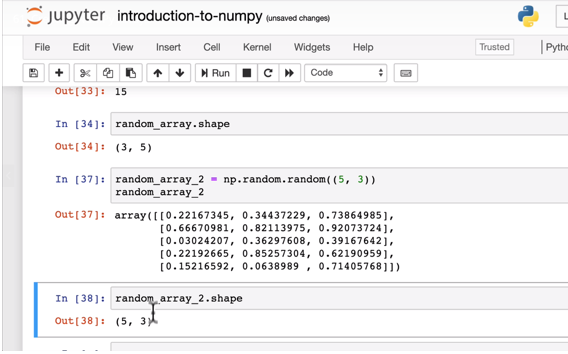
Ones.dtype

ange\_array = np.arange() 🡪 Use shift tab o see what kind of inputs you can give

Random\_array = np.random.randint(0,10,size=(3,5))

Random\_array.size

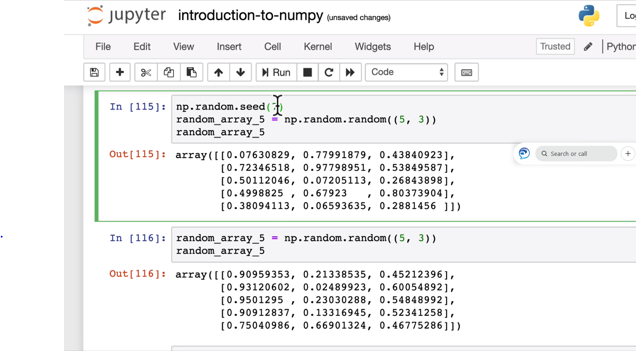
np.random.rand((5,3))



**##pseudo random numbers**

Np.random.seed()

To produce the same set of random numbers irrespective of number of times you run you will get the same random numbers.



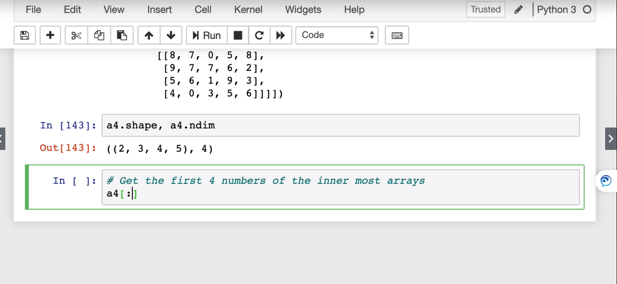
How to find unique numbers in a array?:

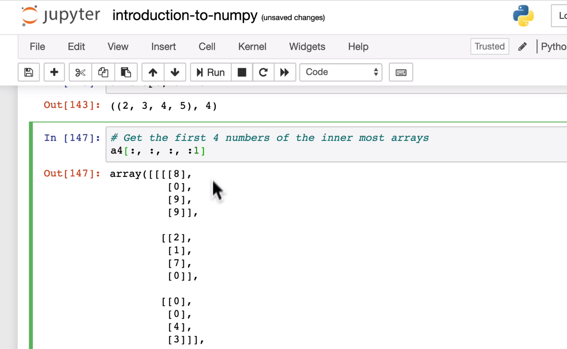
**Numpy array slicing:**

A4 = Np.random.randint(10, size=(2,3,4,5))

A4

A3[:2,:2,:2] 🡪 This will give first 2 columns and rows o fthe 2 dimesnional array.





Manipulating Arrays:

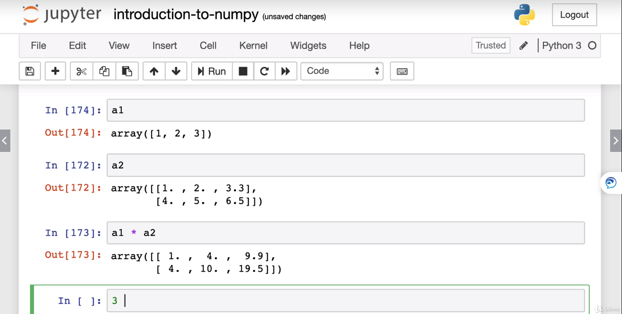
A1

Ones = np.ones(3)

Ones

A1 + ones

A1 \* a2



**Arithmetics:**

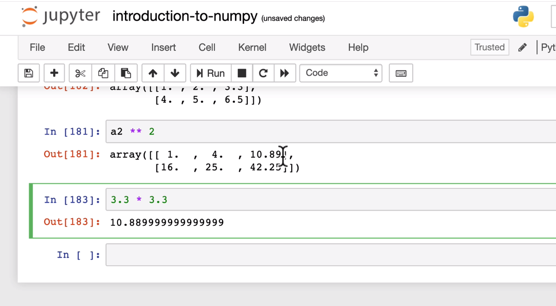
**Floor division rounds of the decimals:**

A1/ones

A2 / a1

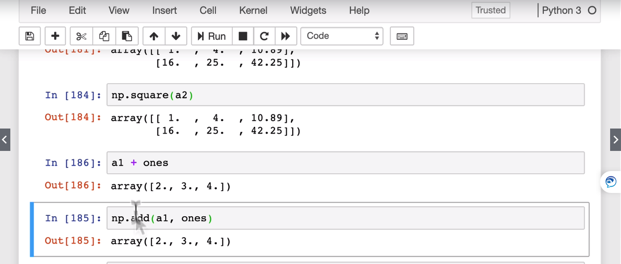
A2 //a1 (removes the decimals)

Square:

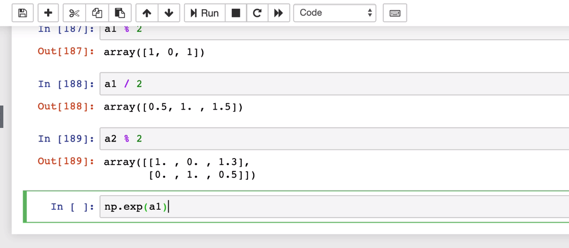


Np.square

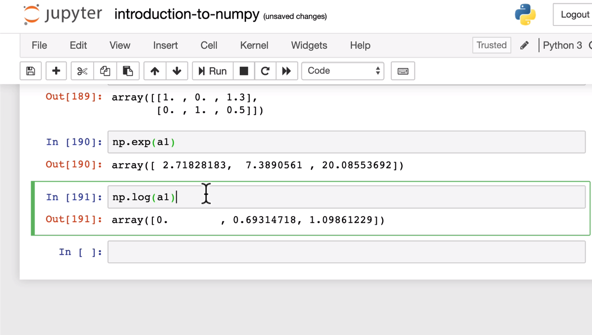
Np.add



Np.exp(a1)



Np.log



Aggregation:

