In [1]:

```
a=10
b=11
print(a,b)
#here a and b are variables which are used to store data as a and b
```

```
10 11
```

In [2]:

```
del b
#it is uded to delete variable
```

In [3]:

```
#%clear #it will clear entire output(console)
#%reset #it will delete the emtire code
```

In [4]:

```
#basic library in python
Numpy-numerical python
pandas-dataframe python
matplpotib-visualisation
sklearn-machine learning
```

```
  File "<ipython-input-4-46a54031cc61>", line 2
    Numpy-numerical python
                         ^
SyntaxError: invalid syntax
```

In [5]:

```python
import numpy
content=dir(numpy)
print(content)
#it will help to print the sub-library of python.
```

```
['ALLOW_THREADS', 'AxisError', 'BUFSIZE', 'CLIP', 'Com
plexWarning', 'DataSource', 'ERR_CALL', 'ERR_DEFAULT',
'ERR_IGNORE', 'ERR_LOG', 'ERR_PRINT', 'ERR_RAISE', 'ER
R_WARN', 'FLOATING_POINT_SUPPORT', 'FPE_DIVIDEBYZERO',
'FPE_INVALID', 'FPE_OVERFLOW', 'FPE_UNDERFLOW', 'False
_', 'Inf', 'Infinity', 'MAXDIMS', 'MAY_SHARE_BOUNDS',
'MAY_SHARE_EXACT', 'MachAr', 'ModuleDeprecationWarnin
g', 'NAN', 'NINF', 'NZERO', 'NaN', 'PINF', 'PZERO', 'R
AISE', 'RankWarning', 'SHIFT_DIVIDEBYZERO', 'SHIFT_INV
ALID', 'SHIFT_OVERFLOW', 'SHIFT_UNDERFLOW', 'ScalarTyp
e', 'Tester', 'TooHardError', 'True_', 'UFUNC_BUFSIZE_
DEFAULT', 'UFUNC_PYVALS_NAME', 'VisibleDeprecationWarn
ing', 'WRAP', '_NoValue', '_UFUNC_API', '__NUMPY_SETUP
__', '__all__', '__builtins__', '__cached__', '__confi
g__', '__dir__', '__doc__', '__file__', '__getattr__',
'__git_revision__', '__loader__', '__mkl_version__',
'__name__', '__package__', '__path__', '__spec__', '__
version__', '_add_newdoc_ufunc', '_distributor_init',
'_globals', '_mat', '_pytesttester', 'abs', 'absolut
e', 'add', 'add_docstring', 'add_newdoc', 'add_newdoc_
ufunc', 'alen', 'all', 'allclose', 'alltrue', 'amax',
'amin', 'angle', 'any', 'append', 'apply_along_axis',
'apply_over_axes', 'arange', 'arccos', 'arccosh', 'arc
sin', 'arcsinh', 'arctan', 'arctan2', 'arctanh', 'argm
ax', 'argmin', 'argpartition', 'argsort', 'argwhere',
'around', 'array', 'array2string', 'array_equal', 'arr
ay_equiv', 'array_repr', 'array_split', 'array_str',
'asanyarray', 'asarray', 'asarray_chkfinite', 'asconti
guousarray', 'asfarray', 'asfortranarray', 'asmatrix',
'asscalar', 'atleast_1d', 'atleast_2d', 'atleast_3d',
'average', 'bartlett', 'base_repr', 'binary_repr', 'bi
ncount', 'bitwise_and', 'bitwise_not', 'bitwise_or',
'bitwise_xor', 'blackman', 'block', 'bmat', 'bool', 'b
ool8', 'bool_', 'broadcast', 'broadcast_arrays', 'broa
dcast_to', 'busday_count', 'busday_offset', 'busdaycal
endar', 'byte', 'byte_bounds', 'bytes0', 'bytes_', 'c
_', 'can_cast', 'cast', 'cbrt', 'cdouble', 'ceil', 'cf
loat', 'char', 'character', 'chararray', 'choose', 'cl
ip', 'clongdouble', 'clongfloat', 'column_stack', 'com
```

```
mon_type', 'compare_chararrays', 'compat', 'complex',
'complex128', 'complex64', 'complex_', 'complexfloatin
g', 'compress', 'concatenate', 'conj', 'conjugate', 'c
onvolve', 'copy', 'copysign', 'copyto', 'core', 'corrc
oef', 'correlate', 'cos', 'cosh', 'count_nonzero', 'co
v', 'cross', 'csingle', 'ctypeslib', 'cumprod', 'cumpr
oduct', 'cumsum', 'datetime64', 'datetime_as_string',
'datetime_data', 'deg2rad', 'degrees', 'delete', 'depr
ecate', 'deprecate_with_doc', 'diag', 'diag_indices',
'diag_indices_from', 'diagflat', 'diagonal', 'diff',
'digitize', 'disp', 'divide', 'divmod', 'dot', 'doubl
e', 'dsplit', 'dstack', 'dtype', 'e', 'ediff1d', 'eins
um', 'einsum_path', 'emath', 'empty', 'empty_like', 'e
qual', 'errstate', 'euler_gamma', 'exp', 'exp2', 'expa
nd_dims', 'expm1', 'extract', 'eye', 'fabs', 'fastCopy
AndTranspose', 'fft', 'fill_diagonal', 'find_common_ty
pe', 'finfo', 'fix', 'flatiter', 'flatnonzero', 'flexi
ble', 'flip', 'fliplr', 'flipud', 'float', 'float16',
'float32', 'float64', 'float_', 'float_power', 'floati
ng', 'floor', 'floor_divide', 'fmax', 'fmin', 'fmod',
'format_float_positional', 'format_float_scientific',
'format_parser', 'frexp', 'frombuffer', 'fromfile', 'f
romfunction', 'fromiter', 'frompyfunc', 'fromregex',
'fromstring', 'full', 'full_like', 'fv', 'gcd', 'gener
ic', 'genfromtxt', 'geomspace', 'get_array_wrap', 'get
_include', 'get_printoptions', 'getbufsize', 'geterr',
'geterrcall', 'geterrobj', 'gradient', 'greater', 'gre
ater_equal', 'half', 'hamming', 'hanning', 'heavisid
e', 'histogram', 'histogram2d', 'histogram_bin_edges',
'histogramdd', 'hsplit', 'hstack', 'hypot', 'i0', 'ide
ntity', 'iinfo', 'imag', 'in1d', 'index_exp', 'indice
s', 'inexact', 'inf', 'info', 'infty', 'inner', 'inser
t', 'int', 'int0', 'int16', 'int32', 'int64', 'int8',
'int_', 'intc', 'integer', 'interp', 'intersect1d', 'i
ntp', 'invert', 'ipmt', 'irr', 'is_busday', 'isclose',
'iscomplex', 'iscomplexobj', 'isfinite', 'isfortran',
'isin', 'isinf', 'isnan', 'isnat', 'isneginf', 'isposi
nf', 'isreal', 'isrealobj', 'isscalar', 'issctype', 'i
ssubclass_', 'issubdtype', 'issubsctype', 'iterable',
'ix_', 'kaiser', 'kron', 'lcm', 'ldexp', 'left_shift',
'less', 'less_equal', 'lexsort', 'lib', 'linalg', 'lin
space', 'little_endian', 'load', 'loads', 'loadtxt',
'log', 'log10', 'log1p', 'log2', 'logaddexp', 'logadde
xp2', 'logical_and', 'logical_not', 'logical_or', 'log
ical_xor', 'logspace', 'long', 'longcomplex', 'longdou
ble', 'longfloat', 'longlong', 'lookfor', 'ma', 'mafro
```

```
mtxt', 'mask_indices', 'mat', 'math', 'matmul', 'matri
x', 'matrixlib', 'max', 'maximum', 'maximum_sctype',
'may_share_memory', 'mean', 'median', 'memmap', 'meshg
rid', 'mgrid', 'min', 'min_scalar_type', 'minimum', 'm
intypecode', 'mirr', 'mkl', 'mod', 'modf', 'moveaxis',
'msort', 'multiply', 'nan', 'nan_to_num', 'nanargmax',
'nanargmin', 'nancumprod', 'nancumsum', 'nanmax', 'nan
mean', 'nanmedian', 'nanmin', 'nanpercentile', 'nanpro
d', 'nanquantile', 'nanstd', 'nansum', 'nanvar', 'nbyt
es', 'ndarray', 'ndenumerate', 'ndfromtxt', 'ndim', 'n
dindex', 'nditer', 'negative', 'nested_iters', 'newaxi
s', 'nextafter', 'nonzero', 'not_equal', 'nper', 'np
v', 'numarray', 'number', 'obj2sctype', 'object', 'obj
ect0', 'object_', 'ogrid', 'oldnumeric', 'ones', 'ones
_like', 'os', 'outer', 'packbits', 'pad', 'partition',
'percentile', 'pi', 'piecewise', 'place', 'pmt', 'pol
y', 'poly1d', 'polyadd', 'polyder', 'polydiv', 'polyfi
t', 'polyint', 'polymul', 'polynomial', 'polysub', 'po
lyval', 'positive', 'power', 'ppmt', 'printoptions',
'prod', 'product', 'promote_types', 'ptp', 'put', 'put
_along_axis', 'putmask', 'pv', 'quantile', 'r_', 'rad2
deg', 'radians', 'random', 'rate', 'ravel', 'ravel_mul
ti_index', 'real', 'real_if_close', 'rec', 'recarray',
'recfromcsv', 'recfromtxt', 'reciprocal', 'record', 'r
emainder', 'repeat', 'require', 'reshape', 'resize',
'result_type', 'right_shift', 'rint', 'roll', 'rollaxi
s', 'roots', 'rot90', 'round', 'round_', 'row_stack',
's_', 'safe_eval', 'save', 'savetxt', 'savez', 'savez_
compressed', 'sctype2char', 'sctypeDict', 'sctypeNA',
'sctypes', 'searchsorted', 'select', 'set_numeric_op
s', 'set_printoptions', 'set_string_function', 'setbuf
size', 'setdiff1d', 'seterr', 'seterrcall', 'seterrob
j', 'setxor1d', 'shape', 'shares_memory', 'short', 'sh
ow_config', 'sign', 'signbit', 'signedinteger', 'sin',
'sinc', 'single', 'singlecomplex', 'sinh', 'size', 'so
metrue', 'sort', 'sort_complex', 'source', 'spacing',
'split', 'sqrt', 'square', 'squeeze', 'stack', 'std',
'str', 'str0', 'str_', 'string_', 'subtract', 'sum',
'swapaxes', 'sys', 'take', 'take_along_axis', 'tan',
'tanh', 'tensordot', 'test', 'testing', 'tile', 'timed
elta64', 'trace', 'tracemalloc_domain', 'transpose',
'trapz', 'tri', 'tril', 'tril_indices', 'tril_indices_
from', 'trim_zeros', 'triu', 'triu_indices', 'triu_ind
ices_from', 'true_divide', 'trunc', 'typeDict', 'typeN
A', 'typecodes', 'typename', 'ubyte', 'ufunc', 'uint',
'uint0', 'uint16', 'uint32', 'uint64', 'uint8', 'uint
```

```
c', 'uintp', 'ulonglong', 'unicode', 'unicode_', 'unio
n1d', 'unique', 'unpackbits', 'unravel_index', 'unsign
edinteger', 'unwrap', 'use_hugepage', 'ushort', 'vande
r', 'var', 'vdot', 'vectorize', 'version', 'void', 'vo
id0', 'vsplit', 'vstack', 'warnings', 'where', 'who',
'zeros', 'zeros_like']
```

If you want to to use the sublibrary of any library we can use in this way libraryname.sublibraryname. for example- numpy.lib

In [6]:

```
#1)values assigned to variable using an assigment operator '='
#2) variable name should be short and desccriptive
#3) avoid one character variable names because one character variable r
```

In [7]:

```
Age=55
age=56
age2=57
Age2=58
print(Age)
print(age)
print(age2)
print(Age2)
#the first letter must start with alphabet but it do not start with any
```

```
55
56
57
58
```

In [8]:

```
employe_id=99
#only underscore is used as special character it can used at begining c
#any other special character used in variable will throw an error
```

In [9]:

```
# If we give the variable name one by one and then give the value in ar
physics,chemistry,math=89,90,75
print(physics)
print(chemistry)
print(math)
```

```
89
90
75
```

```
Types of data types
data types            valuse                         representation
a)boolean           true and false                         bool
b)integer            set of all integers                   int
c)complex             set of all complex numbers           complex
d)float                 floating point number              float
e)string                  sequence of character            str
```

In [10]:

```
#statically typed languahe-----
#1.type of variable is known at compile time
#2-- type of variable declare upfront
#3-- eg java,c++
#--------------------------------------------------------
#dynamically typed language
#1--- type of variable known at run time
#2---variable type need not be declared
#3+--- python,php.
```

identifying object data type:- 1-type(object)#object can be variable or array or tuple or list

In [11]:

```
employee_name="ram"
age=55
height=10.56


type(height)
#gives the data type of height
```

Out[11]:

```
float
```

In [12]:

```
type(age)#gives the data type of age
```

Out[12]:

```
int
```

In [13]:

```
type(employee_name)#gives the data type of employee_name
```

Out[13]:

```
str
```

```
verifying the object data type
1-verify if an object is of certain data type
2-type(object) is datatype
```

In [14]:

```
type(height) is int
#output is always be boolean
```

Out[14]:

```
False
```

In [15]:

```
 type(age) is float
```

Out[15]:

```
False
```

In [16]:

```
type(employee_name) is str
```

Out[16]:

```
True
```

```
Coericing object to a new data type
1-convert the data type of an object to another object
2+-Syntax:datatype(object)
3-changes can be stored in same variable or in diffrent variable
```

In [17]:

```
type(height)
```

Out[17]:

```
float
```

In [18]:

```
ht=int(height)#storing the int value of height in ht
type(ht)
```

Out[18]:

```
int
```

In [19]:

```
#only few coerins are accepted
#1)consider the variable 'salary_tier' which is of string data type
#2)salary tier contains an integer enclosed between the single quotes
```

In [20]:

```
salary_tier='1'
type(salary_tier)
```

Out[20]:

```
str
```

In [21]:

```
salary_tier=int(salary_tier)
```

In [22]:

```
type(salary_tier)
```

Out[22]:

```
int
```

In [23]:

```
#however if the value enclosed within quotes is a string then conversic
```

In [24]:

```
employee_name="ram"
employee_name =float(employee_name)#it will show error because the word
```

```
---------------------------------------------------------
-------------------
ValueError                              Traceback (mos
t recent call last)
<ipython-input-24-bae77f040e45> in <module>
      1 employee_name="ram"
----> 2 employee_name =float(employee_name)#it will show
error because the word can not convert

ValueError: could not convert string to float: 'ram'
```

```
Diffrent type of operator
1)Arthmetic
2)Assigment
3)relational and comparison
```

4)logical
5)bitwise
Operator-- are special symbols that help in carrying out an assigment operation or arthmetic or logical computation
2)value that the operator operates on is called operand

In [ ]:

```
2+3#here + is operator
#2 and 3 are operand
```

Arthmatic operator
1)used to perform mathematical operation between two operands
2)create two variable a and b with values 10 and 5 respectively

In [ ]:

```
a=10
b=5
#SYMBOL=+
#OPERATION=addition
#example
a+b
```

In [ ]:

```
#SYMBOL=-
#OPERATION=substraction
#example
a-b
```

In [ ]:

```
#SYMBOL=*
#OPERATION=multipliction
#example
a*b
```

In [ ]:

```
#SYMBOL=/
#OPERATION=division
#example
a/b
```

In [ ]:

```
#SYMBOL=%
#OPERATION=remainder
#example
a%b
```

In [ ]:

```
#SYMBOL=**
#OPERATION=exponent
#example
a**b
```

```
Decreasing order of precedences
Operations
1)parenthesis
()
2)exponent
**
3)division
/
4)multiplication
*
5)addition and substraction
+,-
```

In [ ]:

```
#ASSigment operator=used to assign values to variables
```

```
Symbol                     operations
example
   (=)                  assign values from right side operand
a=10,b=5
                          to left side operand
 +=                   adds right operand to left operand         a+=b
                       and stores result on left side operand
=15
```

In [ ]:

```
#Relational or comparison operator=tests numerical qualities and inequa
```

In [45]:

```
x=5
y=7
#Symbol <
#operation=strictly less than
#example
print(x<y)
```

```
True
```

In [46]:

```
#Symbol <=
#operation=less than equal to
#example
print(x<=y)
```

```
True
```

In [47]:

```
#Symbol ==
#operation=equal to equal to
#example
print(x==y)
```

```
False
```

In [48]:

```
#Symbol !=
#operation=not equal to
#example
print(x!=y)
```

```
True
```

Logical operator = used when operands are conditional statements and returns boolean value

```
2)In python,LOGICAL operators are designed to work with scalars or
boolean values
```

```
Symbol-or
operations=logical or
example:
```

In [49]:

```python
print((x>y)or(x<y))
```

```
True
```

```
Symbol-and
operations=logical and
example:
```

In [50]:

```python
print((x>y) and (x<y))
```

```
False
```

```
Symbol-not
operations=logical and
example:
```

In [51]:

```python
print(not(x==y))#it generally changes the output answers
```

```
True
```

```
Bitwise operator
1)used when operands are integers
2)integers are treated as string of binary digits
3)operators are bit by bit
4)can also operate on conditional statements which compare scalar
values or arrays
5)bitwise or(|),and(&)
```

In [52]:

```
#create two variable x and y with values 5 and 7 respectively
#binary code for 5 is 0000 0101 and for 7 is 0000 0111
#0 correspond to false and 1 correspond to true
```

In [53]:

```
#acording to binary 5=00000101 and 7=00000111
#since the value of 5 and 7 mathes at 000000111 by comparing each (in b
x=5
y=10
print((x<y)|(x==y))#here the first condition is true so the result is c
```

```
True
```

```
Decreasing order of paranthesis
operation
1)parenthesis                                                       ()
2)exponent **
3)division
/
4)multiplication
*
5)addition and substraction
+,-
6)bitwise and
  &
7)bitwise or
   |
8)relational/comparision operator                             ==,!=,>,>=,
<.<=
9)logical not
not
10)logical and
and
11)logical or                                                       or
```

```
Lists
1)generic data structure in python consisting of an ordered
collection of objects
2)objects in a list are also known as elements or componenets
3)elments of a list need not be of same data type
4)elements of a list need not be of same data type
5)enclosed between two square brackets
```

In [54]:

```
#1)create a list employee id and names
#2)create a variable that contains the number of employees
id=[1,2,3,4]
employee_name=["ram","preeti","satish","john"]
#it is seperated by comma
num_emp=4
```

In [55]:

```
#create an employee list using employee id,employee name and number of
employee_list=[id,employee_name,num_emp]
#to view list
print(employee_list)
```

```
[[1, 2, 3, 4], ['ram', 'preeti', 'satish', 'john'], 4]
```

Indexing in list
1)there are two types of indexing-posistive and negative
2)positive indexing
a)starts from left most indexing
b)0 is the first indexing

In [56]:

```
employee_name=["ram","preeti","satish","john"]
#ram has index 0
#preeti has  index 1 et
#this is called positive indexing
```

3)negative indexing
a)starts from right most element
b)-1 is the first index

In [57]:

```
employee_name=["ram","preeti","satish","john"]
#john has index -1
#satish has index -2
#preeti has index -3
#ram has index -4
```

Accesing components of a list
1)to acess top level components use slicing operator []
2)for sub level /inner level components use [] followed by another []

In [58]:

```
employee_list=[id,employee_name,num_emp]#here we the index of id is 0
#we want to see employee_name
print(employee_list[1])
#to extract id from employee id
print(employee_list[0])
#to extract preeti from the level employee_name that belongs to employe
print(employee_list[1][1])#we give it 1 because the index of preeti is
#to extract the second id from the level id that belongs to the employe
print(employee_list[0][1])
```

```
['ram', 'preeti', 'satish', 'john']
[1, 2, 3, 4]
preeti
2
```

Modifying components of a list
1)elements inside a list can be modified using two methods
2)assigning the new element directly to the index position that has to be updated
3)using in built function where the element that is to be updated with it is given as an input to the function along with the index position

In [59]:

```
#modifying the component of a list using index
#assign the values to be changed to the coresponding index of list
#1)change the value of top level component of a list
employee_list=[id,employee_name,num_emp]
#here we have to update the value of 4 to 5
employee_list[2]=5
print(employee_list)
#change the value of sub level component of a list
#we have to change john to karan
employee_list[1][3]="karan"
print(employee_list)
```

```
[[1, 2, 3, 4], ['ram', 'preeti', 'satish', 'john'], 5]
[[1, 2, 3, 4], ['ram', 'preeti', 'satish', 'karan'], 5]
```

Modifying the component using append():-
append()-adds an objct at the end of the list
Syntax:list_name[index].append(object)
in the above syntax if the index is not specified then the object gets added as a new level in the existing list

There are two ways to add an object to a list:
1)adding an element of a list
2)adding a list to a list this is called concation of a list

In [60]:

```
#1)adding an element to a list
#2)adding number 5 to the level id in employee_list
employee_list[0].append(5)
print(employee_list)
```

```
[[1, 2, 3, 4, 5], ['ram', 'preeti', 'satish', 'karan'],
5]
```

In [61]:

```
#add name nirmal to the level employee_name in employee_list
employee_list[1].append("nirmal")
print(employee_list)
```

```
[[1, 2, 3, 4, 5], ['ram', 'preeti', 'satish', 'karan',
'nirmal'], 5]
```

In [62]:

```
#adding a list to list
#adding a new list age to the existing employee_list
age=[23,24,36,43,52]
employee_list.append(age)
print(employee_list)#it run two times by mistake so it is two times add
```

```
[[1, 2, 3, 4, 5], ['ram', 'preeti', 'satish', 'karan',
'nirmal'], 5, [23, 24, 36, 43, 52]]
```

In [63]:

```
modfying the component using insert()
Syntax:list_name[index].insert(position,object)
```

```
  File "<ipython-input-63-61ace9f699de>", line 1
    modfying the component using insert()
               ^
SyntaxError: invalid syntax
```

In [64]:

```
#adding number '6' at the first position to the level id from the emplo
employee_list[0].insert(0,6)
print(employee_list)
```

```
[[6, 1, 2, 3, 4, 5], ['ram', 'preeti', 'satish', 'kara
n', 'nirmal'], 5, [23, 24, 36, 43, 52]]
```

```
removing the element from the list using del command
del-removes the object at the specified index number
```

syntax:del list_name[index1][index2]
in the above syntax:-
1)index1-index number of the top level components to be droped
2)index2-corresponds to the sub level of the component to be droped

In [65]:

```
#drop the level i.e age from the list
del employee_list[3]
print(employee_list)
```

```
[[6, 1, 2, 3, 4, 5], ['ram', 'preeti', 'satish', 'kara
n', 'nirmal'], 5]
```

In [ ]:

```

```

modifying the component using remove()
1)remove()-removes the first matching object from a list
2)syntax:-list_name[index].remove(object)

In [66]:

```
employee_list[1].remove("ram")#it will delete the first ocuuring of ra
print(employee_list)
```

```
[[6, 1, 2, 3, 4, 5], ['preeti', 'satish', 'karan', 'nirm
al'], 5]
```

In [67]:

```
salary=["high","low","medium","low"]
salary.remove("low")
print(salary)
```

```
['high', 'medium', 'low']
```

modifying the component from the list using pop()
pop()-displays the object that is being removed from the list at the speified index number
Syntax:-list_name[index1].pop(index2)
index1-index number of the top level of the component to be droped

```
index2-corespond to the sub level component to be droped
```

In [68]:

```
#removing 4 from the 5th position of level id from tthe employee_list
employee_list[0].pop(4)#it is preinting which element we are deleting
```

Out[68]:

```
4
```

In [69]:

```
print(employee_list)
```

```
[[6, 1, 2, 3, 5], ['preeti', 'satish', 'karan', 'nirma
l'], 5]
```

```
Tuples
1)consists of ordered collection of objects
2)some of the operations on tuple are similar to lists
3)tuples are enclosed between parnthesis()
4)immuutable=once created they can not be modified
```

In [70]:

```
#Creating the tuples
#create a tuple with employyee id,name,age,salary
employee_details=('P001','JOHN',35,40000)
print (employee_details)
```

```
('P001', 'JOHN', 35, 40000)
```

```
INDEXING
1)POSITIVE INDEXING STARTS FROM 0
2)NEGATIVE INDEXING STARTS FROM -1
```

In [71]:

```
#TO ACESS COMPONENTS OF A TUPLE
#1)TO ACESS COMPONENTS USE SINGE SLICING OPERATOR []
#Syntax:tupl_name[index]
```

In [72]:

```
#to extract id from employee_details
print(employee_details[0])
```

```
P001
```

In [73]:

```
#to etract salary from employee details
print(employee_details[3])
```

```
40000
```

In [74]:

```
#index value out of three gives the eroor out of rangr
```

Slicing
1)used to acess a set of element from a tuple by creating a range of index numbers [x:y]
1)x-index number is where the slice starts(inclusive)
2)y-index number is where the slice ends (exclusive)
3)elements are extracted from x to y-1

In [75]:

```
#to extract name and age from the employee_details
print(employee_details[1:3])#here only 1 and 2 index number are added
```

```
('JOHN', 35)
```

In [76]:

```
#to extract id,employee name,age from the employee_details
print(employee_details[:3])#it means the printing is start from index
```

```
('P001', 'JOHN', 35)
```

LENGTH OF A TUPLE
len()-returns the length of a tuple

```
len(tuple_name)
```

In [77]:

```python
len(employee_details)
```

Out[77]:

```
4
```

```
finding minimum and maximum from tuple
IT is appplicable when we integer or float value
1)min returns the value which is smallest
2)max return the value which is greatest
Syntax:min(tuple_name)
```

In [78]:

```python
english=(56,85,96,75,12)
print(english)
```

```
(56, 85, 96, 75, 12)
```

In [79]:

```python
min(english)
```

Out[79]:

```
12
```

In [80]:

```python
max(english)
```

Out[80]:

```
96
```

```
combining two tuples
1)two tuples can be concatenated as follows
(tuple1)+(tuple2) we can combine more tuples similarly
```

In [81]:

```
#create the two tuples and combine them
employee_details=('pp01','john',35,40000)
employee_details2=('m.com','accounts')
print(employee_details+employee_details2)
```

```
('pp01', 'john', 35, 40000, 'm.com', 'accounts')
```

In [ ]:

```

```

```
Modifying components of a tuple
1)tuples are diffrent from lists in sense tuples can not be modified
2)elements can not be added or remove from tuples using index number
or function(append,del,remove,etc)
```

In [82]:

```
employee_details[0]='pp02'#it shows error if it do not change the eleme
```

```
---------------------------------------------------------
------------------
TypeError                                 Traceback (mos
t recent call last)
<ipython-input-82-258e86220fc8> in <module>
----> 1 employee_details[0]='pp02'#it shows error if it
 do not change the element in the the tuple

TypeError: 'tuple' object does not support item assignme
nt
```

Dctionary 1)python dictionaries is an example of hash table data structure 2)works like key-value pairs ,where the keys are mapped to values 3)dictionaries are enclosed by curly braces{} keys values petrol 1 diesel 2

In [83]:

```
#create the dictionary with diffrent fuel type category
fuel_type={"petrol":1,"diesel":2,"cng":3}
#petrol,diesel and cng are keys hence they are immutable
#values and keys are of any data type
print(fuel_type)
```

```
{'petrol': 1, 'diesel': 2, 'cng': 3}
```

Accesing components of dictionary

In [84]:

```
#to know the value of the key petrol from the fuel_type
print(fuel_type['petrol'])
```

```
1
```

In [85]:

```
#to acess the keys from dictonary fuel_type
#syntax:dictonary_name.keys()
fuel_type.keys()
```

Out[85]:

```
dict_keys(['petrol', 'diesel', 'cng'])
```

In [86]:

```
#to acess the value from dictonary
#Syntax:dictonary_name.values()
fuel_type.values()
```

Out[86]:

```
dict_values([1, 2, 3])
```

In [87]:

```
#to acess both keys and values simultaneously from dictonary
#syntax:-dictonary_name.items()
fuel_type.items()#it return elements in a list format with (key,value)
```

Out[87]:

```
dict_items([('petrol', 1), ('diesel', 2), ('cng', 3)])
```

Dictonary is mutable we can modify components in dictonary
1)Adding new key value pair to the existing dictonary fuel_type using keys
Syntax dictonary_name[key]=value

In [88]:

```
fuel_type['electric']=4
print(fuel_type)
```

```
{'petrol': 1, 'diesel': 2, 'cng': 3, 'electric': 4}
```

In [89]:

```
Adding a new key value pair to the existing dictonary fuel_type using u
syntax:-dictonary_name.update({key:value})
```

```
  File "<ipython-input-89-240bd980e0b3>", line 1
    Adding a new key value pair to the existing dictonar
y fuel_type using update() function
         ^
SyntaxError: invalid syntax
```

In [90]:

```
fuel_type.update({'nuclear':5})
print(fuel_type)#it will delete the first element pair also
```

```
{'petrol': 1, 'diesel': 2, 'cng': 3, 'electric': 4, 'nuc
lear': 5}
```

Modify the value of existing key
1)Assign the value to be changed to corresponding key to dictonary

In [91]:

```
fuel_type['cng']=6
print(fuel_type)#it will change the value of cng
```

```
{'petrol': 1, 'diesel': 2, 'cng': 6, 'electric': 4, 'nuc
lear': 5}
```

Modify dictonary using del()
del- removes the key value pairs
syntax:- del dictonary_name[key]

In [92]:

```
#drop the key from the petrol type
del fuel_type['diesel']
print(fuel_type)
```

```
{'petrol': 1, 'cng': 6, 'electric': 4, 'nuclear': 5}
```

In [93]:

```
#clear it will remove all the dictonary values from dictonary
fuel_type.clear()
print(fuel_type)
```

```
{}
```

Sets
1)set is a cccollection of distinct object
2)it do not hold duplicate items
3)stores the element in no particular order
4)created by curly braces{}

In [94]:

```
#create the set
age={56,52,41,63,41}
print(age)#her the duplicate  value of 41 is deleted and the order is
```

```
{56, 41, 52, 63}
```

In [95]:

```
#order is diffrent as given in question and ouput
employee_name={'Ram','satish','preethi','john','nirmal'}
print (employee_name)
```

```
{'nirmal', 'john', 'Ram', 'satish', 'preethi'}
```

```
Modify the set using add()
1)add()-add element to the existing set at any position
2)add the ganesh to the existing set employee_name
Syntax:-set_name.add(object)
```

In [96]:

```
#add ganesh to the set
employee_name.add('ganesh')
print(employee_name)
```

```
{'nirmal', 'ganesh', 'john', 'Ram', 'satish', 'preethi'}
```

```
Modify the set using discard
1)Discard()-removes the matching object from an existing set
Syntax:- set_name.discard(object)
```

In [97]:

```
#drop the john from the set
employee_name.discard('john')
print(employee_name)
```

```
{'nirmal', 'ganesh', 'Ram', 'satish', 'preethi'}
```

In [98]:

```
#clear will delete all the element present in the set
employee_name.clear()
print(employee_name)
```

```
set()
```

In [99]:

```
junior={'R','python','tableau'}
data={'R','python','scala','java','tableau'}
```

```
Union- returns all the elements present in both a and b
Syntax-set_A.union(set_B)
```

In [100]:

```
union=junior.union(data)
print(union)
```

```
{'scala', 'python', 'java', 'R', 'tableau'}
```

```
Intersection()- returns the elements common to set A and B
Syntax-set_A.intersection(set_b)
```

In [101]:

```
intersection=junior.intersection(data)
print(intersection)
```

```
{'R', 'python', 'tableau'}
```

```
diffrence()-returns elements beleonging to Abut not b
Syntax-set_A.diffrence(set_B)
```

In [102]:

```
diff=junior.difference(data)
print(diff)#because no element left in A after get sub from b
```

```
set()
```

```
Symetric_diffrence
1)symetric_diffrence()-returns elements not common to both sets
Syntax:set_A.symmetric_diffrence(set_B)
```

In [103]:

```
sym_diff=junior.symmetric_difference(data)
print(sym_diff)
```

```
{'java', 'scala'}
```

```
NUMPY
1)numpy stands for numerical python
2)fundamental package for numerical computation in python means for
adding and subtracting
3)supports N-dimensional array objects that can be used for
pprocessing multi-dimensional data.
4)suppports diffrent data type
```

In [104]:

```
#USING NUMPY WE CAN perform
#1) mathematical and logical operation on arrays
#2) fourier tansforms
#3) linear algebra operaion
#4) random number generation
```

```
Create an array
1)orderd collection of element oof basic data type of given length
2)Syntax:numpy.array(object)
```

In [105]:

```
import numpy as np
x=np.array([2,3,4,5])#we are creating an array by numpy
print(x)
```

```
[2 3 4 5]
```

In [106]:

```
 print(type(x))#here we are get that it is numpy and dimensional array
```

```
<class 'numpy.ndarray'>
```

In [107]:

```
#numpy can have diffrent data types elements but it will print all the
x=np.array([2,3,'n',5])#it can handle the element of diffrent type
print(x)#but it will print the output in only one data type
```

```
['2' '3' 'n' '5']
```

In [108]:

```
 #WE can generate arrays randomly
```

```
Generating arrays using linspace()
a)numpy.linspace()-returns equally spaced numbers within the given
range based on sample numbers
b)Syntax-numpy.linspace(start,stop,num,dtype,restep)
c)start-start the interval range
d)stop-end of interval range
e)num-number of the samples genrated
f)dtype-type of output array
g)restep-return the samples ,step value
```

In [109]:

```
b=np.linspace(start=1,stop=5,num=10,endpoint=True,retstep=False)
print(b)
#endpoint=true means fie is incluude
#retstep =false means it will return the samples not incriment values
#if we do not give endpoint!=true then the fie wouls not include at las
```

```
[1.         1.44444444 1.88888889 2.33333333 2.77777778
3.22222222
 3.66666667 4.11111111 4.55555556 5.        ]
```

In [110]:

```
#let us takke restep =true
b=np.linspace(start=1,stop=5,num=10,endpoint=True,retstep=True)
print(b)#the incremented value also get printed means how much value is
```

```
(array([1.        , 1.44444444, 1.88888889, 2.33333333,
2.77777778,
       3.22222222, 3.66666667, 4.11111111, 4.55555556,
5.        ]), 0.4444444444444444)
```

```
Genrate the arrays using arrange()
1)numpy.arrange()=returns equally spaced number with in the given
range based in step size
2)Syntax:numpy.arrange(start,stop,step)
3)start-start of the interval range
4)step-step size of intervaal
```

In [111]:

```
#generate an array with start=1 and step=10 by specifying step=2
d=np.arange(start=1,stop=10,step=2)
print(d)#all is get incremented  by 2
```

```
[1 3 5 7 9]
```

```
numppy.ones()-returns an array of given shape and type filled with
ones
1)Syntax:numpy.ones(shape,dtype)
2)shape=integer orr sequencee of integer
3)dtype=data type(default:float) if we do not specify dtype it is
taken as one
```

In [112]:

```
np.ones((3,4))
#it means 3 rows and 4 columns with default dtype as float
```

Out[112]:

```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [113]:

```
np.zeros((3,4))
#it will print arry 3,4 all filled with zeroes
```

Out[113]:

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

Suppose we want to generate random numbers then we can use random.rand()
Generate arrays using random.rand()
1)numpy.random.rand()-returns an array of givn shape filled with random values
2)Syntax:numpy.random.rand(shape)
shape=how much random numbers we want to genrate it can be inteeger or sequence of integer

In [114]:

```
np.random.rand(5)# we are generating 5 random values here
```

Out[114]:

```
array([0.04394587, 0.60039774, 0.33691235, 0.01711423,
0.18322335])
```

In [115]:

```
#generate an array of random values with 5 rows and 2 columns
np.random.rand(5,2)
```

Out[115]:

```
array([[0.07314009, 0.38665067],
       [0.9609083 , 0.68340315],
       [0.41203162, 0.32718855],
       [0.575755  , 0.35660932],
       [0.11404866, 0.31072692]])
```

Generating arrays using logs

1)numpy.logspace()=returns equally spaced number based on log scale
Syntax:-numpy.logspace(start,stop,num,endpoint,base,dtype)
1)start-start the value of sequence
2)stop-end the value of sequence
3)num-number of samples genrated(default:50)
4)endpoint-if true,stop is the last sample

In [116]:

```
#geneerate an array of 5 samples with base 10
np.logspace(1,10,num=5,endpoint=True,base=10.0)
```

Out[116]:

```
array([1.00000000e+01, 1.77827941e+03, 3.16227766e+05,
5.62341325e+07,
       1.00000000e+10])
```

Advantages of numpy
1)Numpy supports vectorized operation
2)Array operation are carried out in C and hence the universal functions in numpy are faster than operation are carried out in python list

In [117]:

```
#Timeit
```

timeit-module can be used to measure the execution time for snippets of code
2)comparring the processing speed of a list and array using an additional operation

In [118]:

```
#creating a list and calculating its time
x=range(1000)
timeit sum(x)
```

```
  File "<ipython-input-118-800cb602581e>", line 3
    timeit sum(x)
             ^
SyntaxError: invalid syntax
```

In [119]:

```
#creating a numpy and calculating its time
y=np.array(x)
timeit np.sum(y)
#hence the numpy is faster
```

```
  File "<ipython-input-119-fb24bc87ef51>", line 3
    timeit np.sum(y)
             ^
SyntaxError: invalid syntax
```

```
a)getsizeof()-returns the size of the object in bytes
Synntax:sys.getsizeof(object)
1)sys-it is inbuilt function it is use for system specific parameter
b)itemsize-returns the size of one eleement of a numpy array
Syntax-numpy.ndarray.itemsize
```

In [120]:

```
#we can compare the size of list and numpy by using getsizeof
```

In [121]:

```
#we are calculating the size for the list
#1) Size of list can be found by multiplying the size of an individual
import sys
sys.getsizeof(1)*len(x)
```

Out[121]:

```
56
```

In [122]:

```
#size of an array can be found by multiplying the size of an individual
y.itemsize * y.size
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-122-e78849ca9f1b> in <module>
      1 #size of an array can be found by multiplying the size of an individual element eith the number of element in the array
----> 2 y.itemsize * y.size

AttributeError: 'int' object has no attribute 'itemsize'
```

In [123]:

```
#storage space is less in numpy as compare to list
```

Reshaping an array
reshape()- recasts an array to a new shape without changing its data

In [124]:

```
import numpy as np
grid=np.arange(start=1,stop=10).reshape(3,3)
print(grid)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [125]:

```
#we can also create 3*3 array as shown in figure
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(a)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [126]:

```
#if we want to know the dimensions of an arrAy
#1)reeturns the dimensions of an array
#2)Syntax:array_name.shape
a.shape
#it will give dimensions
```

Out[126]:

```
(3, 3)
```

```
Numpy addition
1)numpy.add()-performs elementwise addition bbetween two arrays
2)Syntaz:numpy.add(array_1,array_2)
3)shape of two array should be same
```

In [127]:

```
#create two array a and b then add it
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(a)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [128]:

```
b=np.arange(start=11,stop=20).reshape(3,3)#create two array through
print(b)
```

```
[[11 12 13]
 [14 15 16]
 [17 18 19]]
```

In [129]:

```
np.add(a , b)#it will help to add a and b acording to their matching in
```

Out[129]:

```
array([[12, 14, 16],
       [18, 20, 22],
       [24, 26, 28]])
```

numpy.multiply()-performs elementwise operation between two arrays
Syntax:numpy.multiply(array_1,array_2)

In [130]:

```
np.multiply(a,b)#it generally multiply and b element
```

Out[130]:

```
array([[ 11,  24,  39],
       [ 56,  75,  96],
       [119, 144, 171]])
```

1)numpy.substract-performs element wise substraction between two arrays
2)numpy.divide-returns the element wise division of inputs
3)numpy.remainder-return the element wise remainder of division

In [131]:

```
#Accessing the component of an array
#a)components of an array can be accessed using index numbers
```

In [132]:

```
 #extract element with index (0,1) from a
a[0,1]
```

Out[132]:

```
2
```

In [133]:

```
#extraxt all the element from  2 and 3 row
a[1:3]
```

Out[133]:

```
array([[4, 5, 6],
       [7, 8, 9]])
```

In [134]:

```
#extract elements from the first column of array a
a[:,0]#: it means sellect all the rows 0 means first column select all
```

Out[134]:

```
array([1, 4, 7])
```

In [135]:

```
#to select the first row of array
a[0:1]
#or
a[0,:]
```

Out[135]:

```
array([1, 2, 3])
```

```
Subset of an array
1)Subset a 2*2 array from the orginal array a
2)consider the first two rows and columns from a
```

In [136]:

```
#2*2 subset from a
a_subset=a[:2,:2]#we are giving this command becuse in before comma we
print(a_subset)
```

```
[[1 2]
 [4 5]]
```

In [137]:

```
#3*2 ubset from a
z=a[:3,:2]
print(z)
```

```
[[1 2]
 [4 5]
 [7 8]]
```

In [138]:

```
 #now we wnat to change the value of 1 to 12 in subset
a_subset[0,0]=12
print(a_subset)
```

```
[[12  2]
 [ 4  5]]
```

In [139]:

```
#modifying the subset will also change the main set
print(a)#value of 1 change to 12
```

```
[[12  2  3]
 [ 4  5  6]
 [ 7  8  9]]
```

Modifying array using transpose()
1)numpy.transpose()-change the column into the row and row into column
Syntax:-numpy.transpose(array)

In [140]:

```
print(a)
```

```
[[12  2  3]
 [ 4  5  6]
 [ 7  8  9]]
```

In [141]:

```
np.transpose(a)
```

Out[141]:

```
array([[12,  4,  7],
       [ 2,  5,  8],
       [ 3,  6,  9]])
```

Modify array using append()
append()-adds the value at the end of the array
Syntax:numpy.append(array,axis)

In [142]:

```
a_row=np.append(a,[[10,11,14]],axis=0)
print(a_row)#it means we are  adding an row at last
```

```
[[12  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 14]]
```

In [143]:

```
#adding an array column wise
col =np.array([21,22,23]).reshape(3,1)#we are creating an array of 3 ro
print(col)
```

```
[[21]
 [22]
 [23]]
```

In [144]:

```
a_col=np.append(a,col,axis=1)#we are adding a column to existing array
print(a_col)
```

```
[[12  2  3 21]
 [ 4  5  6 22]
 [ 7  8  9 23]]
```

insert()-adds value at a given position and axis in an array

1)if we give acess is equal to 1 it add column wise but if you give value axis=0 then the addition would be happen row wise
1)Syntax:numpy.insert(array,obj,values,axis)
a)array-input array
b)obj-index position
c)values-array of values to be inserted
d)is-axis along which values should be inserted

In [145]:

```
#insert new array along row nd the 1st index position
a_ins=np.insert(a,1,[13,15,16],axis=0)#a is array name we want to inser
print(a_ins)
```

```
[[12  2  3]
 [13 15 16]
 [ 4  5  6]
 [ 7  8  9]]
```

delete()-it removes values at a given position and axis in an array
Syntax:-numpy.delete(array,obj,axis)
1)array-input array
2)obj-indicate array to be removed or its posittion
3)axis along which array should be removed

In [146]:

```
#delete third row from th exissting array a_ins
a_del=np.delete(a_ins,2,axis=0)
print(a_del)
```

```
[[12  2  3]
 [13 15 16]
 [ 7  8  9]]
```

MATRICES
1)rectangular arrangement of numbers in rows and columns
2)rows run horizontally and columns run vertically
3)3*3 means three rows and three columns
4)3*1 means 3 rows and 1 columns
5)1*3 means 1 rows amd 3 columns

In [147]:

```
#create a matrix
```

matrix()-returns a matrix from an array type object or string of data
Syntax:- numpy.matrix(data)

In [148]:

```
import numpy as np
a=np.matrix("1,2,3,4;4,5,6,7;7,8,9,10")
print(a)
```

```
[[ 1  2  3  4]
 [ 4  5  6  7]
 [ 7  8  9 10]]
```

Matrix properties
1)shape()-returns the number of rows and columns from a matrix

In [149]:

```
a.shape
```

Out[149]:

```
(3, 4)
```

a)shape[0]-returns the number of rows
b)shape[1]-returns the number of columns

In [150]:

```
a.shape[0]#it gives how many rows are present in a matrix
```

Out[150]:

```
3
```

In [151]:

```
a.shape[1]#it gives how many columns are prresent in columns are preser
```

Out[151]:

```
4
```

```
size()-returns the number of element from a matrix
```

In [152]:

```
a.size
```

Out[152]:

```
12
```

MODIFY MATRIX USING INSERT()

```
insert-adds values at a given position and axis in am matrix
Syntax:-numpy.insert(matrix,obj,values,axis)
1)matrix-input matrix
2)obj-index position
3)values-matrix of values to be inserted
4)axis-axis along which values should be inserted
```

In [153]:

```
print(a)
```

```
[[ 1  2  3  4]
 [ 4  5  6  7]
 [ 7  8  9 10]]
```

In [154]:

```
col_new=np.matrix("2,3,4")#creating the new matrix so that we can add 1
print(col_new)
```

```
[[2 3 4]]
```

In [155]:

```
import numpy as np
a=np.insert(a,0,col_new,axis=1)
#1)a rwprent where we want to insert the new matrix
#we give axis=1 because we want to enter column_wise
print(a)#hence the col_new added to zeroth position
```

```
[[ 2  1  2  3  4]
 [ 3  4  5  6  7]
 [ 4  7  8  9 10]]
```

Adding the matrix 'row_new' as a new row to a

In [156]:

```
row_new=np.matrix("4,5,6,7,9")
print(row_new)
```

```
[[4 5 6 7 9]]
```

In [157]:

```
a=np.insert(a,0,row_new,axis=0)
print(a)#row is added first
```

```
[[ 4  5  6  7  9]
 [ 2  1  2  3  4]
 [ 3  4  5  6  7]
 [ 4  7  8  9 10]]
```

Modifying matrix using index
1)elements of a matrix can be modified using index number

In [158]:

```
#here the value 1 should be updated to -3
a[1,1]=-3
#becuse we have to change -1 to 3 and it lies on first row and first co
print(a)
```

```
[[ 4  5  6  7  9]
 [ 2 -3  2  3  4]
 [ 3  4  5  6  7]
 [ 4  7  8  9 10]]
```

Accesing the element of matrix using index or slicing

```
Extract the element from second row of matrix a
```

In [159]:

```
#ectract second row from matrix
print(a[1,:])#it means select all columns from row 1
```

```
[[ 2 -3  2  3  4]]
```

In [160]:

```
#extract elements from third column of matrix a
print(a[:,2])
```

```
[[6]
 [2]
 [5]
 [8]]
```

In [161]:

```
#extract element from index(1,2) from a
print(a[1,2])
```

```
2
```

Matrix addition

```
numpy.add()=performs elementwise addition between two matrices
Syntax:numpy.add(matrix_1,matrix_2)
```

In [162]:

```
#create two matrices A and B
A=np.matrix(np.arange(0,20)).reshape(5,4)
B=np.matrix(np.arange(20,40)).reshape(5,4)
```

In [163]:

```
print(A)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

In [164]:

```
print(B)
```

```
[[20 21 22 23]
 [24 25 26 27]
 [28 29 30 31]
 [32 33 34 35]
 [36 37 38 39]]
```

In [165]:

```
print(np.add(A,B))#it will add element wise
```

```
[[20 22 24 26]
 [28 30 32 34]
 [36 38 40 42]
 [44 46 48 50]
 [52 54 56 58]]
```

```
Matrix substraction
1)numpy.subtract()-performs elementwise substraction between two
matrices
Syntax:numpy.substract(matrix_1,matrix_2)
```

```
#for element wise substraction we use this
np.subtract(A,B)#it is performing A-B
```

```
Matrix multiplication()
1)numpy.dot()-performs matrix multiplication between two matrices
Syntax:numpy.dot(matrix_1,matrix_2)
```

In [166]:

```
#multiply A and B
#it will multiply first row and first column
np.dot(A,B)
#it show error because we do not follow rule of column of matrix a shou
```

```
---------------------------------------------------------
-------------------
ValueError                                  Traceback (mos
t recent call last)
<ipython-input-166-0cc98bf19550> in <module>
      1 #multiply A and B
      2 #it will multiply first row and first column
----> 3 np.dot(A,B)
      4 #it show error because we do not follow rule of
 column of matrix a should equal to row of matrix b

<__array_function__ internals> in dot(*args, **kwargs)

ValueError: shapes (5,4) and (5,4) not aligned: 4 (dim
 1) != 5 (dim 0)
```

In [167]:

```
#now we are transposing matrix B to get 4*5 dimension
B=np.transpose(B)
np.dot(A,B)
```

Out[167]:

```
matrix([[ 134,  158,  182,  206,  230],
        [ 478,  566,  654,  742,  830],
        [ 822,  974, 1126, 1278, 1430],
        [1166, 1382, 1598, 1814, 2030],
        [1510, 1790, 2070, 2350, 2630]])
```

numpy.multiply()-performs element wise multiplication between two matrices
Syntax:numpy.multiply(matrix_1,matrix_2)

In [168]:

```
np.multiply(A,B)
```

```
---------------------------------------------------------
-------------------
ValueError                                Traceback (mos
t recent call last)
<ipython-input-168-53549bd6e8c7> in <module>
----> 1 np.multiply(A,B)

ValueError: operands could not be broadcast together wit
h shapes (5,4) (4,5)
```

In [169]:

```
numpy.divide()-performs elementwise division betwwn two matrix
Syntax-numpy.divide(matrix_1,matrix_2)
```

```
  File "<ipython-input-169-b5ac51767e5c>", line 1
    numpy.divide()-performs elementwise division betwwn
 two matrix
                              ^
SyntaxError: invalid syntax
```

In [170]:

```
np .divide(A,B)#HENCE IT WILL PERFORM DIVISION WHEN THE DIMENSION RULE
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-170-2d9f1f233e4f> in <module>
----> 1 np .divide(A,B)#HENCE IT WILL PERFORM DIVISION WHEN THE DIMENSION RULE FOLLOWED

ValueError: operands could not be broadcast together with shapes (5,4) (4,5)
```

LINEAR ALGEBRA

```
Determinant of matrix
1)matrix should be square matrix
2)numpy.linalg.det()-returns the determinant of matrix
```

In [171]:

```
x=np.matrix("4,5,16,7;2,-3,2,3;3,4,5,6;4,7,8,9")
#we have created 4*4 matrix
```

In [172]:

```
print(x)
```

```
[[ 4  5 16  7]
 [ 2 -3  2  3]
 [ 3  4  5  6]
 [ 4  7  8  9]]
```

In [173]:

```
det_matrix=np.linalg.det(x)#it will find out ndeterminent of matrix
print(det_matrix)
```

```
128.00000000000009
```

```
1)numpy.linalg.matrix_rank()-returns rank of matrix
Syntax:numpy.linalg.matrix_rank(matrix)
```

In [174]:

```
rank_matrix=np.linalg.matrix_rank(x)
print(rank_matrix)
#it will print rank of matrix
#rank 4 means it has four independent row
```

```
4
```

```
Inverse of a matrix
1)numpy.linalg.inv()-returns the multiplicative inverse of a matrix
2)Syntax:numpy.linalg.inv(matrix)
```

In [175]:

```
#creating a matrix A
A=np.matrix("3,1,2;3,2,5;6,7,9")
print(A)
```

```
[[3 1 2]
 [3 2 5]
 [6 7 9]]
```

In [176]:

```
inv_matrix=np.linalg.inv(A)
#it will print the inverse of matrix
print(inv_matrix)
```

```
[[ 0.56666667 -0.16666667 -0.03333333]
 [-0.1        -0.5         0.3       ]
 [-0.3         0.5        -0.1       ]]
```

In [177]:

```
#create a matrix B
B=np.matrix("2,1,1;1,0,1;3,1,3")
print(B)
```

```
[[2 1 1]
 [1 0 1]
 [3 1 3]]
```

In [178]:

```
inverse_matrix=np.linalg.inv(B)
print(inverse_matrix)
#it is a singular matrix hence its inverse is not posiible singular mea
```

```
[[ 1.  2. -1.]
 [ 0. -3.  1.]
 [-1. -1.  1.]]
```

In [179]:

```
#find the determinent of B
deter_matrix=np.linalg.det(B)
print(deter_matrix)
```

```
-1.0
```

System of linear equation

1)if we have two or more linear equation then it is called as system of linear equation
2)if we solve two or more equation we can get unique solution or no solution or infinetly many solution

In [180]:

```
#solving linar equation
```

```
consider a system of linear equation
3x+y+2z=2
3x+2y+5z=-1
6x+7y+8z=3
Now we can write it Ax=b
3x+y+2z=2
3x+2y+5z=-1   -------------------->[3,1,2        [X
6x+7y+8z=3                          4,2,5          Y  =[2,-1,3].
                                    6,7,8].        Z]     b
                                    A               X
```

In [181]:

```
#numpy.linalg.solve()-return the solution to system AX=b
#Syntax:numpy.linalg.solve(matrix_A,matrix_b)
#create matrix A and b
A=np.matrix("3,1,2;3,2,5;6,7,8")
print(A)
```

```
[[3 1 2]
 [3 2 5]
 [6 7 8]]
```

In [182]:

```
b=np.matrix("2,1,3").transpose()#this has done to change the row and co
print(b)
```

```
[[2]
 [1]
 [3]]
```

In [183]:

```
sol_linear=np.linalg.solve(A,b)
print(sol_linear)
#this is the value of x,y and z
```

```
[[ 0.87878788]
 [ 0.09090909]
 [-0.36363636]]
```

Reading data

```
File format
1)Standard way in which data is collected and stored
2)most commonly used format for storing data is the spreadsheet
format where data is stored in rows and columns
3)eaxh row is called a record
4)Each column in an spreadsheet holds data belonging to same data
type
5)commonly used spreadsheet format are comma spreated values and
excel sheet
6)other format include plain text,json,html,mp3.mp4 etc.
```

In [184]:

```
#csv format
```

```
CSV-COMMA SEPERATED VALUE
1)SPREADSHEET FORMAT
2)Format is '.csv' means if we save something with csv it become csv format
3)Each record is seprated by a
4)files where records are seprated using a tab are called tab seprated values
5).csv file can also be opened from notepad or microsoft excel
6)if you open csv file in notepad the problem is we cannot diffrentiate between the row and column and which cell belong to which variable
```

In [185]:

```
#excel spreadsheet
```

```
1)it is aspreadsheet format
2)part of microsoft office
3)format'_xlsx' if we save something with '._xlsx' it become excel sheet
```

In [186]:

```
#TEXT FRMAT
```

```
1)CONSISTS OF plain text or records
2)format '.txt'
```

In [187]:

```
#IMPORT INTO SPYDER
```

```
1)IMPORTING NECESSARY FILE IN DATA
2)import os----'os'library to change the working directory
3)import pandas as pd------"pandas" libraray to work with data frame.whever we read data in python it beconme data frame
4)each data is represented in tabular form where each row is represented by sample and each column as variable
```

represented by sample and each column as variable

In [188]:

```
import pandas as pd
#changing the working directory
#os.chidr("H:\")#os.chidr represent the changing directory in semicolun
#then give file name from which we want to extract data
data_csv=pd.read_csv('Iris_data_sample.csv')#data_csv is data frame
#pd.read helps us to read the data
#.csv in compulsary
print(data_csv)
```

```
     sepal_length  sepal_width  petal_length  petal_widt
h    species
0             5.1          3.5           1.4           0.
2     setosa
1             4.9          3.0           1.4           0.
2     setosa
2             4.7          3.2           1.3           0.
2     setosa
3             4.6          3.1           1.5           0.
2     setosa
4             5.0          3.6           1.4           0.
2     setosa
..            ...          ...           ...
...        ...
145           6.7          3.0           5.2           2.
3  virginica
146           6.3          2.5           5.0           1.
9  virginica
147           6.5          3.0           5.2           2.
0  virginica
148           6.2          3.4           5.4           2.
3  virginica
149           5.9          3.0           5.1           1.
8  virginica

[150 rows x 5 columns]
```

1)in data frame all blank cells read as 'nan'

In [189]:

```
#for making index of any column
data_csv=pd.read_csv('Iris_data_sample.csv',index_col=0)
print(data_csv)#here we are making first line as index
```

```
              sepal_width  petal_length  petal_width
species
sepal_length
5.1                   3.5           1.4          0.2
setosa
4.9                   3.0           1.4          0.2
setosa
4.7                   3.2           1.3          0.2
setosa
4.6                   3.1           1.5          0.2
setosa
5.0                   3.6           1.4          0.2
setosa
...                   ...           ...          ...
...
6.7                   3.0           5.2          2.3  vi
rginica
6.3                   2.5           5.0          1.9  vi
rginica
6.5                   3.0           5.2          2.0  vi
rginica
6.2                   3.4           5.4          2.3  vi
rginica
5.9                   3.0           5.1          1.8  vi
rginica

[150 rows x 4 columns]
```

In [190]:

```
#if you want to convert all the special character with nan values then
data_csv=pd.read_csv('Iris_data_sample.csv',index_col=0,na_values=["??"
#here all double ?? converted to na_values
data_csv=pd.read_csv('Iris_data_sample.csv',index_col=0,na_values=["??"
#here all ?? and ## converted to nan values
```

In [191]:

```
#importing excel file
import pandas as pd
data_xlsx=pd.read_excel('Iris_data_sample.xlsx')
#this is how we import import excel file
```

```
---------------------------------------------------------
-------------------
XLRDError                                 Traceback (mos
t recent call last)
<ipython-input-191-1d076cd92244> in <module>
      1 #importing excel file
      2 import pandas as pd
----> 3 data_xlsx=pd.read_excel('Iris_data_sample.xlsx')
      4 #this is how we import import excel file

~\anaconda3\lib\site-packages\pandas\util\_decorators.py
in wrapper(*args, **kwargs)
    294                 )
    295                 warnings.warn(msg, FutureWarning
, stacklevel=stacklevel)
--> 296             return func(*args, **kwargs)
    297 
    298         return wrapper

~\anaconda3\lib\site-packages\pandas\io\excel\_base.py i
n read_excel(io, sheet_name, header, names, index_col, u
secols, squeeze, dtype, engine, converters, true_values,
false_values, skiprows, nrows, na_values, keep_default_n
a, na_filter, verbose, parse_dates, date_parser, thousan
ds, comment, skipfooter, convert_float, mangle_dupe_col
s)
    302 
    303     if not isinstance(io, ExcelFile):
--> 304         io = ExcelFile(io, engine=engine)
    305     elif engine and engine != io.engine:
    306         raise ValueError(

~\anaconda3\lib\site-packages\pandas\io\excel\_base.py i
n __init__(self, path_or_buffer, engine)
    865         self._io = stringify_path(path_or_buffer
)
    866 
--> 867         self._reader = self._engines[engine](sel
f._io)
    868
    869     def __fspath__(self):

~\anaconda3\lib\site-packages\pandas\io\excel\_xlrd.py i
n __init__(self, filepath_or_buffer)
     20         err_msg = "Install xlrd >= 1.0.0 for Exc
el support"
     21         import_optional_dependency("xlrd", extra
=err_msg)
---> 22         super().__init__(filepath_or_buffer)
     23
     24     @property

~\anaconda3\lib\site-packages\pandas\io\excel\_base.py i
n __init__(self, filepath_or_buffer)
    351             self.book = self.load_workbook(filep
ath_or_buffer)
    352         elif isinstance(filepath_or_buffer, str)
:
--> 353             self.book = self.load_workbook(filep
ath_or_buffer)
    354         elif isinstance(filepath_or_buffer, byte
s):
    355             self.book = self.load_workbook(Bytes
IO(filepath_or_buffer))

~\anaconda3\lib\site-packages\pandas\io\excel\_xlrd.py i
n load_workbook(self, filepath_or_buffer)
     35             return open_workbook(file_contents=d
ata)
     36         else:
---> 37             return open_workbook(filepath_or_buf
fer)
     38
     39     @property

~\anaconda3\lib\site-packages\xlrd\__init__.py in open_w
orkbook(filename, logfile, verbosity, use_mmap, file_con
tents, encoding_override, formatting_info, on_demand, ra
gged_rows)
    146
    147     from . import book
--> 148     bk = book.open_workbook_xls(
    149         filename=filename,
    150         logfile=logfile,
~\anaconda3\lib\site-packages\xlrd\book.py in open_workb
ook_xls(filename, logfile, verbosity, use_mmap, file_con
tents, encoding_override, formatting_info, on_demand, ra
gged_rows)
     90         t1 = perf_counter()
     91         bk.load_time_stage_1 = t1 - t0
---> 92         biff_version = bk.getbof(XL_WORKBOOK_GLO
BALS)
     93         if not biff_version:
     94             raise XLRDError("Can't determine fil
e's BIFF version")

~\anaconda3\lib\site-packages\xlrd\book.py in getbof(sel
f, rqd_stream)
   1276             bof_error('Expected BOF record; met
 end of file')
   1277         if opcode not in bofcodes:
-> 1278             bof_error('Expected BOF record; foun
d %r' % self.mem[savpos:savpos+8])
   1279         length = self.get2bytes()
   1280         if length == MY_EOF:

~\anaconda3\lib\site-packages\xlrd\book.py in bof_error
(msg)
   1270
   1271         def bof_error(msg):
-> 1272             raise XLRDError('Unsupported format,
or corrupt file: ' + msg)
   1273         savpos = self._position
   1274         opcode = self.get2bytes()

XLRDError: Unsupported format, or corrupt file: Expected
BOF record; found b'sepal_le'
```

In [192]:

```
#importing data as text format
data_txt1=pd.read_table('iris_data_sample.txt')#this how we extract tex
print(data_txt1)
```

```
    sepal_length,sepal_width,petal_length,petal_width,sp
ecies
0                               5.1,3.5,1.4,0.2,setosa
1                               4.9,3.0,1.4,0.2,setosa
2                               4.7,3.2,1.3,0.2,setosa
3                               4.6,3.1,1.5,0.2,setosa
4                               5.0,3.6,1.4,0.2,setosa
..                                                 ...
145                          6.7,3.0,5.2,2.3,virginica
146                          6.3,2.5,5.0,1.9,virginica
147                          6.5,3.0,5.2,2.0,virginica
148                          6.2,3.4,5.4,2.3,virginica
149                          5.9,3.0,5.1,1.8,virginica

[150 rows x 1 columns]
```

1)it is showing 150 rows and 1 columns but we havr 5 variable it is shoeing because all columns read and stored in a single column of a data frame 2)in orrder to avoid this problem provide a delimeter to the parameter 'sep' or 'delimeter'

In [193]:

```
#default delimeter is tab represented by '/t'
data_text1=pd.read_table('Iris_data_sample.txt',sep='/t')
print(data_text1)
```

```
    sepal_length,sepal_width,petal_length,petal_width,sp
ecies
0                                5.1,3.5,1.4,0.2,setosa
1                                4.9,3.0,1.4,0.2,setosa
2                                4.7,3.2,1.3,0.2,setosa
3                                4.6,3.1,1.5,0.2,setosa
4                                5.0,3.6,1.4,0.2,setosa
..                                                  ...
145                           6.7,3.0,5.2,2.3,virginica
146                           6.3,2.5,5.0,1.9,virginica
147                           6.5,3.0,5.2,2.0,virginica
148                           6.2,3.4,5.4,2.3,virginica
149                           5.9,3.0,5.1,1.8,virginica

[150 rows x 1 columns]
```

```
C:\Users\user\anaconda3\lib\site-packages\pandas\io\pars
ers.py:765: ParserWarning: Falling back to the 'python'
engine because the 'c' engine does not support regex sep
arators (separators > 1 char and different from '\s+' ar
e interpreted as regex); you can avoid this warning by s
pecifying engine='python'.
  return read_csv(**locals())
```

In [194]:

```
data_text1=pd.read_table('Iris_data_sample.txt',delimiter='/t')
print(data_text1)
```

```
    sepal_length,sepal_width,petal_length,petal_width,sp
ecies
0                                5.1,3.5,1.4,0.2,setosa
1                                4.9,3.0,1.4,0.2,setosa
2                                4.7,3.2,1.3,0.2,setosa
3                                4.6,3.1,1.5,0.2,setosa
4                                5.0,3.6,1.4,0.2,setosa
..                                                  ...
145                           6.7,3.0,5.2,2.3,virginica
146                           6.3,2.5,5.0,1.9,virginica
147                           6.5,3.0,5.2,2.0,virginica
148                           6.2,3.4,5.4,2.3,virginica
149                           5.9,3.0,5.1,1.8,virginica

[150 rows x 1 columns]
```

```
C:\Users\user\anaconda3\lib\site-packages\pandas\io\pars
ers.py:765: ParserWarning: Falling back to the 'python'
engine because the 'c' engine does not support regex sep
arators (separators > 1 char and different from '\s+' ar
e interpreted as regex); you can avoid this warning by s
pecifying engine='python'.
  return read_csv(**locals())
```

in both tab delimiter is not working

In [195]:

```
data_text1=pd.read_table('Iris_data_sample.txt',delimiter="  ")
print(data_text1)#now it will show 6 columns
```

```
    sepal_length,sepal_width,petal_length,petal_width,sp
ecies
0                               5.1,3.5,1.4,0.2,setosa
1                               4.9,3.0,1.4,0.2,setosa
2                               4.7,3.2,1.3,0.2,setosa
3                               4.6,3.1,1.5,0.2,setosa
4                               5.0,3.6,1.4,0.2,setosa
..                                                 ...
145                          6.7,3.0,5.2,2.3,virginica
146                          6.3,2.5,5.0,1.9,virginica
147                          6.5,3.0,5.2,2.0,virginica
148                          6.2,3.4,5.4,2.3,virginica
149                          5.9,3.0,5.1,1.8,virginica

[150 rows x 1 columns]
```

```
C:\Users\user\anaconda3\lib\site-packages\pandas\io\pars
ers.py:765: ParserWarning: Falling back to the 'python'
engine because the 'c' engine does not support regex sep
arators (separators > 1 char and different from '\s+' ar
e interpreted as regex); you can avoid this warning by s
pecifying engine='python'.
  return read_csv(**locals())
```

In [196]:

```
data_text1=pd.read_csv('Iris_data_sample.txt',delimiter="  ")
print(data_text1)#by using csv we can read text file also by using prop
```

```
    sepal_length,sepal_width,petal_length,petal_width,sp
ecies
0                               5.1,3.5,1.4,0.2,setosa
1                               4.9,3.0,1.4,0.2,setosa
2                               4.7,3.2,1.3,0.2,setosa
3                               4.6,3.1,1.5,0.2,setosa
4                               5.0,3.6,1.4,0.2,setosa
..                                                 ...
145                          6.7,3.0,5.2,2.3,virginica
146                          6.3,2.5,5.0,1.9,virginica
147                          6.5,3.0,5.2,2.0,virginica
148                          6.2,3.4,5.4,2.3,virginica
149                          5.9,3.0,5.1,1.8,virginica

[150 rows x 1 columns]
```

```
<ipython-input-196-af41226b97a5>:1: ParserWarning: Falli
ng back to the 'python' engine because the 'c' engine do
es not support regex separators (separators > 1 char and
different from '\s+' are interpreted as regex); you can
avoid this warning by specifying engine='python'.
  data_text1=pd.read_csv('Iris_data_sample.txt',delimite
r="  ")
```

Pandas

```
Introduction to pandas
1) provides high performance easy to use data structureand analysis
tool for the python programing language
2)open source python library providing high performance data
manipulation and analysis tool uding its powerful data structure
3)Name pandas is derived from the word panel-data an ecconometrics
term for multidimensional data
```

In [197]:

```
#pandas deal with data frame
```

```
Pandas deal with data frame
1)the data frame consist of two dimension first is row and second is
column hence they are two dimensional and size mutable
 2)dataframe is collection of data in tabular form data is arranged
in rows and columns in which rows shows sample or record and column
represent the variable means property associated with each sample
3)potentially hetrogenus tabular data structure with labelled axes
4)hetrogenus table data structure means when ever we read data into
spyder it becomes data frame and each data types get orginal data
associated with that
5)labelled axis means each row and column are labelled for row we
have index and name of column is name of each variable
```

In [198]:

```
#importing data
import pandas as pd
car_data=pd.read_csv('ToyotaCorolla.csv')
print(car_data)
```

```
        Id
Model  Price  \
0         1      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-
Doors  13500
1         2      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-
Doors  13750
2         3     ?TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-
Doors  13950
3         4      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-
Doors  14950
4         5        TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-
Doors  13750
...     ...
...     ...
1431  1438         TOYOTA Corolla 1.3 16V HATCHB G6 2/3-
Doors   7500
1432  1439  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/
3-...  10845
1433  1440  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/
3-...   8500
1434  1441  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/
3-...   7250
1435  1442        TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-
Doors   6950

      Age_08_04  Mfg_Month  Mfg_Year     KM Fuel_Type
HP  Met_Color  ...  \
0            23         10      2002  46986    Diesel
90          1  ...
1            23         10      2002  72937    Diesel
90          1  ...
2            24          9      2002  41711    Diesel
90          1  ...
3            26          7      2002  48000    Diesel
90          0  ...
4            30          3      2002  38500    Diesel
90          0  ...
...         ...        ...       ...    ...       ...
...         ...  ...
```

```
1431         69          12       1998  20544    Petrol
86          1  ...
1432         72           9       1998  19000    Petrol
86          0  ...
1433         71          10       1998  17016    Petrol
86          0  ...
1434         70          11       1998  16916    Petrol
86          1  ...
1435         76           5       1998      1    Petrol  1
10          0  ...

      Central_Lock  Powered_Windows  Power_Steering  Rad
io  Mistlamps  \
0                1                1               1
0          0
1                1                0               1
0          0
2                0                0               1
0          0
3                0                0               1
0          0
4                1                1               1
0          1
...            ...              ...             ...
...        ...
1431             1                1               1
0          1
1432             0                0               1
0          0
1433             0                0               1
0          0
1434             0                0               0
0          0
1435             0                0               1
0          0

      Sport_Model  Backseat_Divider  Metallic_Rim  Radio
_cassette  Tow_Bar
0               0                 1             0
0        0
1               0                 1             0
0        0
2               0                 1             0
0        0
3               0                 1             0
0        0
```

```
4                0                  1             0
0        0
...             ...                ...           ...
...      ...
1431             1                  1             0
0        0
1432             1                  1             0
0        0
1433             0                  1             0
0        0
1434             0                  1             0
0        0
1435             0                  0             0
0        0

[1436 rows x 37 columns]
```

In [199]:

```
#if we want to first column as index
cars=pd.read_csv('ToyotaCorolla.csv',index_col=0)#here 0 is treated wh
print(cars)#if we put 1 here age is treated as index
#if we make something as index its data do not change
```

```
                                                  Model
Price  Age_08_04  \
Id
1         TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors
13500         23
2         TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors
13750         23
3        ?TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors
13950         24
4         TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors
14950         26
5           TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors
13750         30
...                                                 ...
...        ...
1438         TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors
7500         69
1439  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...
10845         72
1440  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...
8500         71
1441  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...
7250         70
1442        TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors
6950         76

      Mfg_Month  Mfg_Year     KM Fuel_Type   HP  Met_Col
or  Automatic  ...  \
Id
...
1            10      2002  46986    Diesel   90
1          0  ...
2            10      2002  72937    Diesel   90
1          0  ...
3             9      2002  41711    Diesel   90
1          0  ...
4             7      2002  48000    Diesel   90
0          0  ...
5             3      2002  38500    Diesel   90
```

```
0          0  ...
...         ...       ...    ...       ...  ...
...        ...  ...
1438         12      1998  20544    Petrol   86
1          0  ...
1439          9      1998  19000    Petrol   86
0          0  ...
1440         10      1998  17016    Petrol   86
0          0  ...
1441         11      1998  16916    Petrol   86
1          0  ...
1442          5      1998      1    Petrol  110
0          0  ...

      Central_Lock  Powered_Windows  Power_Steering  Rad
io  Mistlamps  \
Id
1                1                1               1
0          0
2                1                0               1
0          0
3                0                0               1
0          0
4                0                0               1
0          0
5                1                1               1
0          1
...            ...              ...             ...
...        ...
1438             1                1               1
0          1
1439             0                0               1
0          0
1440             0                0               1
0          0
1441             0                0               0
0          0
1442             0                0               1
0          0

      Sport_Model  Backseat_Divider  Metallic_Rim  Radio
_cassette  Tow_Bar
Id
1               0                 1             0
0        0
2               0                 1             0
```

```
0        0
3               0                  1             0
0        0
4               0                  1             0
0        0
5               0                  1             0
0        0
...           ...                ...           ...
...      ...
1438            1                  1             0
0        0
1439            1                  1             0
0        0
1440            0                  1             0
0        0
1441            0                  1             0
0        0
1442            0                  0             0
0        0

[1436 rows x 36 columns]
```

In [200]:

```
cars_data=pd.read_csv('ToyotaCorolla.csv')
print(cars_data)
```

```
        Id
Model  Price  \
0        1      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/
3-Doors  13500
1        2      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/
3-Doors  13750
2        3     ?TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/
3-Doors  13950
3        4      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/
3-Doors  14950
4        5        TOYOTA Corolla 2.0 D4D HATCHB SOL 2/
3-Doors  13750
...    ...
...    ...
1431  1438         TOYOTA Corolla 1.3 16V HATCHB G6 2/
3-Doors   7500
1432  1439  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA
2/3-...  10845
1433  1440  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA
2/3-...   8500
1434  1441  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA
2/3-...   7250
1435  1442        TOYOTA Corolla 1.6 LB LINEA TERRA 4/
5-Doors   6950

      Age_08_04  Mfg_Month  Mfg_Year     KM Fuel_Type
HP  Met_Color  ...  \
0             23         10      2002  46986    Diesel
90          1  ...
1             23         10      2002  72937    Diesel
90          1  ...
2             24          9      2002  41711    Diesel
90          1  ...
3             26          7      2002  48000    Diesel
90          0  ...
4             30          3      2002  38500    Diesel
90          0  ...
...         ...        ...       ...    ...       ...
...        ...  ...
1431          69         12      1998  20544    Petrol
86          1  ...
```

```
1432         72          9      1998  19000    Petrol
86          0  ...
1433         71         10      1998  17016    Petrol
86          0  ...
1434         70         11      1998  16916    Petrol
86          1  ...
1435         76          5      1998      1    Petrol
110          0  ...

      Central_Lock  Powered_Windows  Power_Steering  R
adio  Mistlamps  \
0                1                1               1
0          0
1                1                0               1
0          0
2                0                0               1
0          0
3                0                0               1
0          0
4                1                1               1
0          1
...            ...              ...             ...
...        ...
1431             1                1               1
0          1
1432             0                0               1
0          0
1433             0                0               1
0          0
1434             0                0               0
0          0
1435             0                0               1
0          0

      Sport_Model  Backseat_Divider  Metallic_Rim  Rad
io_cassette  Tow_Bar
0               0                 1             0
0        0
1               0                 1             0
0        0
2               0                 1             0
0        0
3               0                 1             0
0        0
4               0                 1             0
0        0
```

```
...            ...               ...           ...
...      ...
1431            1                 1             0
0        0
1432            1                 1             0
0        0
1433            0                 1             0
0        0
1434            0                 1             0
0        0
1435            0                 0             0
0        0

[1436 rows x 37 columns]
```

```
Creating a copy of orginal data
1)shallow copy
2)deep copy
```

In [ ]:

```

```

In [201]:

```
#shallow copy
#samp=cars_data or cars_data=data.copy(deep=false) #code for shallow co
#it only creates a new variable that shre reference of the orginal post
#any change made to a copy of object will be reflected in the orginal c
#means any change in samp will refelect in cars_data
```

```
Deep copy
1)cars_data1=cars_data.copy(deep=True)
1)in case of deep copy a copy of object is copied to another object
with no reference to the orginal
2)any changes made a copy of object will not be reflected in the
orginal object
3)any change in cars_data1 wll not reflect in cars_data
```

Attributes of data

In [202]:

```
cars_data1=cars_data.copy(deep=True)
print(cars_data1)
```

```
        Id
Model  Price  \
0        1      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-
Doors  13500
1        2      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-
Doors  13750
2        3     ?TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-
Doors  13950
3        4      TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-
Doors  14950
4        5        TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-
Doors  13750
...    ...
...    ...
1431  1438         TOYOTA Corolla 1.3 16V HATCHB G6 2/3-
Doors   7500
1432  1439  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/
3-...  10845
1433  1440  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/
3-...   8500
1434  1441  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/
3-...   7250
1435  1442        TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-
Doors   6950

      Age_08_04  Mfg_Month  Mfg_Year     KM Fuel_Type
HP  Met_Color   ...  \
0            23         10      2002  46986    Diesel
90          1   ...
1            23         10      2002  72937    Diesel
90          1   ...
2            24          9      2002  41711    Diesel
90          1   ...
3            26          7      2002  48000    Diesel
90          0   ...
4            30          3      2002  38500    Diesel
90          0   ...
...         ...        ...       ...    ...       ...
...        ...  ...
1431         69         12      1998  20544    Petrol
```

```
86          1  ...
1432         72          9      1998  19000    Petrol
86          0  ...
1433         71         10      1998  17016    Petrol
86          0  ...
1434         70         11      1998  16916    Petrol
86          1  ...
1435         76          5      1998      1    Petrol  1
10          0  ...

      Central_Lock  Powered_Windows  Power_Steering  Rad
io  Mistlamps  \
0                1                1               1
0          0
1                1                0               1
0          0
2                0                0               1
0          0
3                0                0               1
0          0
4                1                1               1
0          1
...            ...              ...             ...
...        ...
1431             1                1               1
0          1
1432             0                0               1
0          0
1433             0                0               1
0          0
1434             0                0               0
0          0
1435             0                0               1
0          0

      Sport_Model  Backseat_Divider  Metallic_Rim  Radio
_cassette  Tow_Bar
0               0                 1             0
0        0
1               0                 1             0
0        0
2               0                 1             0
0        0
3               0                 1             0
0        0
4               0                 1             0
```

```
0        0
...           ...               ...           ...
...      ...
1431            1                 1             0
0        0
1432            1                 1             0
0        0
1433            0                 1             0
0        0
1434            0                 1             0
0        0
1435            0                 0             0
0        0

[1436 rows x 37 columns]
```

In [203]:

```
#to get index of data
cars_data1.index
```

Out[203]:

```
RangeIndex(start=0, stop=1436, step=1)
```

In [204]:

```
#to get columns name of all data frame
cars_data1.columns
```

Out[204]:

```
Index(['Id', 'Model', 'Price', 'Age_08_04', 'Mfg_Month',
'Mfg_Year', 'KM',
       'Fuel_Type', 'HP', 'Met_Color', 'Automatic', 'c
c', 'Doors', 'Cylinders',
       'Gears', 'Quarterly_Tax', 'Weight', 'Mfr_Guarante
e', 'BOVAG_Guarantee',
       'Guarantee_Period', 'ABS', 'Airbag_1', 'Airbag_
2', 'Airco',
       'Automatic_airco', 'Boardcomputer', 'CD_Player',
'Central_Lock',
       'Powered_Windows', 'Power_Steering', 'Radio', 'Mi
stlamps',
       'Sport_Model', 'Backseat_Divider', 'Metallic_Ri
m', 'Radio_cassette',
       'Tow_Bar'],
      dtype='object')
```

In [205]:

```
#to get total number of element from data frame
cars_data1.size
```

Out[205]:

```
53132
```

In [206]:

```
#to get dimension of the data frame
cars_data1.shape
```

Out[206]:

```
(1436, 37)
```

In [207]:

```
#to get memory usage of each column in bytes
cars_data1.memory_usage()
```

Out[207]:

```
Index                  64
Id                  11488
Model                5744
Price               11488
Age_08_04           11488
Mfg_Month           11488
Mfg_Year            11488
KM                  11488
Fuel_Type            5744
HP                  11488
Met_Color           11488
Automatic           11488
cc                  11488
Doors               11488
Cylinders           11488
Gears               11488
Quarterly_Tax       11488
Weight              11488
Mfr_Guarantee       11488
BOVAG_Guarantee     11488
Guarantee_Period    11488
ABS                 11488
Airbag_1            11488
Airbag_2            11488
Airco               11488
Automatic_airco     11488
Boardcomputer       11488
CD_Player           11488
Central_Lock        11488
Powered_Windows     11488
Power_Steering      11488
Radio               11488
Mistlamps           11488
Sport_Model         11488
Backseat_Divider    11488
Metallic_Rim        11488
Radio_cassette      11488
Tow_Bar             11488
dtype: int64
```

In [208]:

```
#to get the number of axes/array dimensions means how many dimensions
cars_data1.ndim#becuse only one row and columns are there
```

Out[208]:

```
2
```

Indexing and selecting data
1)python slicing operator'[.]' and attribute/dot operator '.' are used for indexing
2)provide the quick and easy acces of data structure

In [209]:

```
#dataframe.head([n])
#the function head returns the first n rows from the dataframe
cars_data1.head(6)#by the default head() returns first 5 rows
```

Out[209]:

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	F
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002	46986	
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002	72937	
2	3	? TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002	41711	
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	7	2002	48000	
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002	38500	

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	F
5	6	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	12950	32	1	2002	61000	

6 rows × 37 columns

In [210]:

```
#the function tail returns the last n rows for the object based on posi
cars_data1.tail(6)
```

Out[210]:

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	
1430	1437	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8450	80	1	1998	23
1431	1438	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	12	1998	20
1432	1439	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	9	1998	19
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	10	1998	17
1434	1441	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	11	1998	16

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year
1435	1442	TOYOTA Corolla 1.6 LB LINEA TERRA	6950	76	5	1998

```
To acess a scalar value ,the fastest way is to use the at and iat
method
1)at provide label based scalar lookups
2)iat based integer based lookups
```

In [211]:

```
cars_data1.at[4,'Fuel_Type']#it will take 4th row data from column dat
```

Out[211]:

```
'Diesel'
```

In [212]:

```
cars_data1.iat[5,6]#it rweturns 5 the column and 6 th row
```

Out[212]:

```
61000
```

In [213]:

```
cars_data1.loc[:,'Fuel_Type']
#it will print all rows of fueltype column
```

Out[213]:

```
0       Diesel
1       Diesel
2       Diesel
3       Diesel
4       Diesel
         ...
1431    Petrol
1432    Petrol
1433    Petrol
1434    Petrol
1435    Petrol
Name: Fuel_Type, Length: 1436, dtype: object
```

DATA types

```
1)the way information get store in a dataframe or a python object
affects the data analysis and output of calculation
2)there are two main types of data
numeric and character
3)numeric data types includes integers and floats
for example:integer=10,float=10.5
4)strings are known as objects in pandas which semicolumn  can store
values that contain number and /or characters
for example'category1'
```

In [214]:

```
#Numeric data types
```

```
1)pandas and python uses different names for data types
a)python data type is int and float
b)python data type store the data type as int64 and float64
->'64' simply refers to a memory allocated to store data in each cell
which effectively relates to how many digits it can store in each
cell
->64 bits equivalent to 8 bytes
```

In [ ]:

```

```

In [215]:

```
#why we are concern about memory allocation in each cell?
#ANS-aLLOCATING SPACE AHEAD OF TIME ALLOWS COMPUTER TO OPTIMIZE STORAGE
```

```
in python there are two types of data types that can handle string is
category and object
#category
1)A string variabble consisting only a different values converting a
string variable to a categoricaL  variable will save some money
2)A categorical variable takes on limited fixed number of possible
values if have large number of categories we can ise object
#object
1)the column will be assigned to a object data type when when it has
mixed types,If a column contain nan values pandas will default to
object data type
2)all nan contain diffrent values so it became objecyt data type by
default
3)for string length is not fixed we can put as many element in a
string
```

In [216]:

```
#dtypes returns a series with the data type of each column
#Syntax:datframe.dtypes
cars_data1.dtypes
```

Out[216]:

```
Id                   int64
Model               object
Price                int64
Age_08_04            int64
Mfg_Month            int64
Mfg_Year             int64
KM                   int64
Fuel_Type           object
HP                   int64
Met_Color            int64
Automatic            int64
cc                   int64
Doors                int64
Cylinders            int64
Gears                int64
Quarterly_Tax        int64
Weight               int64
Mfr_Guarantee        int64
BOVAG_Guarantee      int64
Guarantee_Period     int64
ABS                  int64
Airbag_1             int64
Airbag_2             int64
Airco                int64
Automatic_airco      int64
Boardcomputer        int64
CD_Player            int64
Central_Lock         int64
Powered_Windows      int64
Power_Steering       int64
Radio                int64
Mistlamps            int64
Sport_Model          int64
Backseat_Divider     int64
Metallic_Rim         int64
Radio_cassette       int64
Tow_Bar              int64
dtype: object
```

In [259]:

```
#count of unique data types
#get_dtyoe_counts()-returns the count of unique data types in data fram
#Syntax:-Dataframe.get_dtype_count()
cars_data1.get_dtype_counts()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (mos
t recent call last)
<ipython-input-259-19d9853720fe> in <module>
      2 #get_dtyoe_counts()-returns the count of unique
 data types in data frame
      3 #Syntax:-Dataframe.get_dtype_count()
----> 4 cars_data1.get_dtype_count()

~\anaconda3\lib\site-packages\pandas\core\generic.py in
__getattr__(self, name)
   5137             if self._info_axis._can_hold_identif
iers_and_holds_name(name):
   5138                 return self[name]
-> 5139             return object.__getattribute__(self,
name)
   5140 
   5141     def __setattr__(self, name: str, value) -> N
one:

AttributeError: 'DataFrame' object has no attribute 'get
_dtype_count'
```

In [218]:

```
#Select data based on data types
```

```
pandas.dataframe.select_dtypes() returns a subset of the column from
dataframe based on the column dtypes
1)Syntax:Dataframe.select_dtypes(include=none,exclude=none)
2)if we want to select only few datatypes we can use include
3)if we do not want to select any datatypes we use exclude
```

In [219]:

```
cars_data1.select_dtypes(exclude=[object])#here we are excluding object
```

Out[219]:

	Id	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	HP	Me
0	1	13500	23	10	2002	46986	90	
1	2	13750	23	10	2002	72937	90	
2	3	13950	24	9	2002	41711	90	
3	4	14950	26	7	2002	48000	90	
4	5	13750	30	3	2002	38500	90	
...	...	...	...	...	...	...	...	
1431	1438	7500	69	12	1998	20544	86	
1432	1439	10845	72	9	1998	19000	86	
1433	1440	8500	71	10	1998	17016	86	
1434	1441	7250	70	11	1998	16916	86	
1435	1442	6950	76	5	1998	1	110	

1436 rows × 35 columns

```
info()-returns a concise summary of a data frame
1)dattype of index
2)data type of columns
3)countof null values
4)memmory usage
5)Syntax:Dataframe.info()
```

In [220]:

```
cars_data1.info()
#here it is pandas core dataframe and id is represented as int64
#price it has 1436 data entries and 1436 null values and int64 as data
#the propse to get summary is to verify  whether all data types read at
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 37 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Id                1436 non-null   int64 
 1   Model             1436 non-null   object
 2   Price             1436 non-null   int64 
 3   Age_08_04         1436 non-null   int64 
 4   Mfg_Month         1436 non-null   int64 
 5   Mfg_Year          1436 non-null   int64 
 6   KM                1436 non-null   int64 
 7   Fuel_Type         1436 non-null   object
 8   HP                1436 non-null   int64 
 9   Met_Color         1436 non-null   int64 
 10  Automatic         1436 non-null   int64 
 11  cc                1436 non-null   int64 
 12  Doors             1436 non-null   int64 
 13  Cylinders         1436 non-null   int64 
 14  Gears             1436 non-null   int64 
 15  Quarterly_Tax     1436 non-null   int64 
 16  Weight            1436 non-null   int64 
 17  Mfr_Guarantee     1436 non-null   int64 
 18  BOVAG_Guarantee   1436 non-null   int64 
 19  Guarantee_Period  1436 non-null   int64 
 20  ABS               1436 non-null   int64 
 21  Airbag_1          1436 non-null   int64 
 22  Airbag_2          1436 non-null   int64 
 23  Airco             1436 non-null   int64 
 24  Automatic_airco   1436 non-null   int64 
 25  Boardcomputer     1436 non-null   int64 
 26  CD_Player         1436 non-null   int64 
 27  Central_Lock      1436 non-null   int64 
 28  Powered_Windows   1436 non-null   int64 
 29  Power_Steering    1436 non-null   int64 
 30  Radio             1436 non-null   int64 
 31  Mistlamps         1436 non-null   int64 
 32  Sport_Model       1436 non-null   int64 
 33  Backseat_Divider  1436 non-null   int64 
 34   Metallic_Rim       1436 non-null   int64
 35   Radio_cassette     1436 non-null   int64
 36   Tow_Bar            1436 non-null   int64
dtypes: int64(35), object(2)
memory usage: 403.9+ KB
```

In [221]:

```
#if we have the values presen in semicolumns for example '' , "" then i
```

unique()-it is used to find th unique elements of columns
Syntax:numpy.unique(array)

In [222]:

```
import numpy as np
print(np.unique(cars_data1['KM']))
```

```
[     1     15    225 ... 218118 232940 243000]
```

In [223]:

```
print(np.unique(cars_data1['HP']))#UNIQUE ELEMENTS OF HP IS PRESENTED
```

```
[ 69  71  72  73  86  90  97  98 107 110 116 192]
```

In [224]:

```
#IF WE REPLACE '?' WITH NAN VALUE THEN THE VALUE OF NOT NULL VALUE IS D
```

CONVERTING VARIABLE'S DATA TYPE

astype()-method is used to explicitly convert data type from one to another
Synatax:Dataframe.astype(dtype)

In [225]:

```
#converting 'metcolor','Automatic' to object data type
cars_data['Metcolor']=cars_data['Metcolor'].astype('object')#here we ar
cars_data['Automatic']=cars_data['Automatic'].astype('object')
cars_data.info()
```

```
---------------------------------------------------------
-------------------
KeyError                                  Traceback (mos
t recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.p
y in get_loc(self, key, method, tolerance)
   2894                 try:
-> 2895                     return self._engine.get_loc(cast
ed_key)
   2896                 except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngin
e.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngin
e.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.
hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.
hashtable.PyObjectHashTable.get_item()

KeyError: 'Metcolor'

The above exception was the direct cause of the followin
g exception:

KeyError                                  Traceback (mos
t recent call last)
<ipython-input-225-74ff56186fc2> in <module>
      1 #converting 'metcolor','Automatic' to object dat
a type
----> 2 cars_data['Metcolor']=cars_data['Metcolor'].asty
pe('object')#here we are converting metcolor datatype to
object datatype
      3 cars_data['Automatic']=cars_data['Automatic'].as
type('object')
```

```
      4 cars_data.info()

~\anaconda3\lib\site-packages\pandas\core\frame.py in __
getitem__(self, key)
   2900             if self.columns.nlevels > 1:
   2901                 return self._getitem_multilevel(
key)
-> 2902             indexer = self.columns.get_loc(key)
   2903             if is_integer(indexer):
   2904                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.p
y in get_loc(self, key, method, tolerance)
   2895                 return self._engine.get_loc(cast
ed_key)
   2896             except KeyError as err:
-> 2897                 raise KeyError(key) from err
   2898 
   2899         if tolerance is not None:

KeyError: 'Metcolor'
```

category vs object data type

In [ ]:

```

```

```
nbytes()-it is used to get the total bytes consumed by the element of
the column
Syntax:ndarray.nbytes
```

In [226]:

```
#if 'Fuel_Type' is of object data type.
cars_data['Fuel_Type'].nbytes
```

Out[226]:

```
5744
```

In [227]:

```
#if Fuel_type is of category datatype
cars_data['Fuel_Type'].astype('category').nbytes
```

Out[227]:

```
1448
```

from above we knew that if we are dealing with less amount of data we use object but if we deal with large amount of data we use category because object uses large amount of byte as compare to object

In [228]:

```
cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 37 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Id                1436 non-null   int64 
 1   Model             1436 non-null   object
 2   Price             1436 non-null   int64 
 3   Age_08_04         1436 non-null   int64 
 4   Mfg_Month         1436 non-null   int64 
 5   Mfg_Year          1436 non-null   int64 
 6   KM                1436 non-null   int64 
 7   Fuel_Type         1436 non-null   object
 8   HP                1436 non-null   int64 
 9   Met_Color         1436 non-null   int64 
 10  Automatic         1436 non-null   int64 
 11  cc                1436 non-null   int64 
 12  Doors             1436 non-null   int64 
 13  Cylinders         1436 non-null   int64 
 14  Gears             1436 non-null   int64 
 15  Quarterly_Tax     1436 non-null   int64 
 16  Weight            1436 non-null   int64 
 17  Mfr_Guarantee     1436 non-null   int64 
 18  BOVAG_Guarantee   1436 non-null   int64 
 19  Guarantee_Period  1436 non-null   int64 
 20  ABS               1436 non-null   int64 
 21  Airbag_1          1436 non-null   int64 
 22  Airbag_2          1436 non-null   int64 
 23  Airco             1436 non-null   int64 
 24  Automatic_airco   1436 non-null   int64 
 25  Boardcomputer     1436 non-null   int64 
 26  CD_Player         1436 non-null   int64 
 27  Central_Lock      1436 non-null   int64 
 28  Powered_Windows   1436 non-null   int64 
 29  Power_Steering    1436 non-null   int64 
 30  Radio             1436 non-null   int64 
 31  Mistlamps         1436 non-null   int64 
 32  Sport_Model       1436 non-null   int64 
 33  Backseat_Divider  1436 non-null   int64 
 34  Metallic_Rim      1436 non-null   int64 
 35  Radio_cassette    1436 non-null   int64 
 36  Tow_Bar           1436 non-null   int64 
```

```
dtypes: int64(35), object(2)
memory usage: 403.9+ KB
```

In [229]:

```
#replace used to replace a value with desired value
syntax:Dataframe.replace([to_replace,value,........])
```

```
  File "<ipython-input-229-514283ef9b21>", line 2
    syntax:Dataframe.replace([to_replace,valu
e,........])
                                                    ^
SyntaxError: invalid syntax
```

In [231]:

```
cars_data["Fuel_Type"].replace("Diesel","petrol",inplace=True)
#it means we are converting diesel into petrol and inplace=true means e
```

To detect missing values

```
to check the missing values present in each column
Dataframe.isnull.sum()is used
```

In [232]:

```
cars_data.isnull().sum()
#here we are getting the sum of null values
```

Out[232]:

```
Id                  0
Model               0
Price               0
Age_08_04           0
Mfg_Month           0
Mfg_Year            0
KM                  0
Fuel_Type           0
HP                  0
Met_Color           0
Automatic           0
cc                  0
Doors               0
Cylinders           0
Gears               0
Quarterly_Tax       0
Weight              0
Mfr_Guarantee       0
BOVAG_Guarantee     0
Guarantee_Period    0
ABS                 0
Airbag_1            0
Airbag_2            0
Airco               0
Automatic_airco     0
Boardcomputer       0
CD_Player           0
Central_Lock        0
Powered_Windows     0
Power_Steering      0
Radio               0
Mistlamps           0
Sport_Model         0
Backseat_Divider    0
Metallic_Rim        0
Radio_cassette      0
Tow_Bar             0
dtype: int64
```

Control structure

```
1)execute certain commands only certain condition(s) is (are)
satisfied(if-then-else)
2)execute certain command repeadatly and use a certain logic to stop
the iretation (for,while,loop)
```

In [233]:

```python
#if else family of constructs
```

```
if,if else and if-elif are a family of constructs where :
1)A condition is first checked if it is satisfied then operation is
perform
2)if condition is not satisfied code exists construct or moves on to
other option
```

In [234]:

```python
#usage rule
```

```
1)if construct-->                             if expression statement
2)if-else construct:-->                 if expression: statement
                                          else:
                                          statement


3)if-elif-elseconstruct                and if expression:
                                                 statement
                                       elif expression2:
                                            statement
                                           else:
                                           stateememt
```

For loop

```
1)execute the certain condition repeadetly and use a certain logic or
stop the iteration (for loop)
2)for                   for iter in sequence statements
```

While loop

```
A while loop is used when a set of commands are to be executed
depending on a specific condition
2)basically it run as long as the condition is true
when the condition become false it stop its iteration
```

```
while                                       while(condition is stasfied) :
                                                  statatements
```

Example :if else and for loop

```
1)we will create 3 bins frpom the price variable using if Else and
for loop
2)the binned value will be stored as classes in a new column ,'Price
Classs'
```

In [ ]:

```

```

In [235]:

```
#HENCE WE ARE CREATING A NEW COLUMN
cars_data1.insert(10,"Price_class","")
```

In [236]:

```
for i in range (0,len(cars_data1['Price']),1):
    if(cars_data1['Price'][i]<=8450):
        cars_data1['Price_class'][i]="Low"
    elif((cars_data1['Price'][i]>119850)):
        cars_data1['Price_Class'][i]="High"
    else: cars_data1['Price_class'][i]="Medium"
```

```
<ipython-input-236-f14495516a1d>:6: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a
DataFrame

See the caveats in the documentation: https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  else: cars_data1['Price_class'][i]="Medium"
<ipython-input-236-f14495516a1d>:3: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a
 DataFrame

See the caveats in the documentation: https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  cars_data1['Price_class'][i]="Low"
```

In [237]:

```
i=0
```

In [238]:

```
while i<len(cars_data1['Price']):
    if(cars_data1['Price'][i]<=8450):
        cars_data1['Price_class'][i]='Low'
    elif((cars_data1['Price'][i]>11950)):
        cars_data1['Price_class'][i]="High"
    else: cars_data1['Price_class'][i]="medium"
    i=i+1#A while loop is used whenver you want to execute statements u
```

```
<ipython-input-238-2079600e916a>:5: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a
DataFrame

See the caveats in the documentation: https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  cars_data1['Price_class'][i]="High"
<ipython-input-238-2079600e916a>:3: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a
 DataFrame

See the caveats in the documentation: https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  cars_data1['Price_class'][i]='Low'
<ipython-input-238-2079600e916a>:6: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a
 DataFrame

See the caveats in the documentation: https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  else: cars_data1['Price_class'][i]="medium"
```

In [ ]:

```python



#series.value_counts() returns series conataining count of unique value
#by using it we can get how many times a values is added tio a particul
```

In [ ]:

```python
cars_data1['Price_class'].value_counts()
#we have converted numerical value into the categorical value hence we
```

Functions in python

```
1)A function accepts input arguments and produce an output by
executing valid commands present in the functions
2) Function name and file name need not be same
3)A file have one or more function defination
4)A function are created using the command def and a colon with the
statement to be executed indented as a block
5)def function_name (parameter):
    statements  this statement will be solved according to the
parameter 2)the statement is followed based on the intention
```

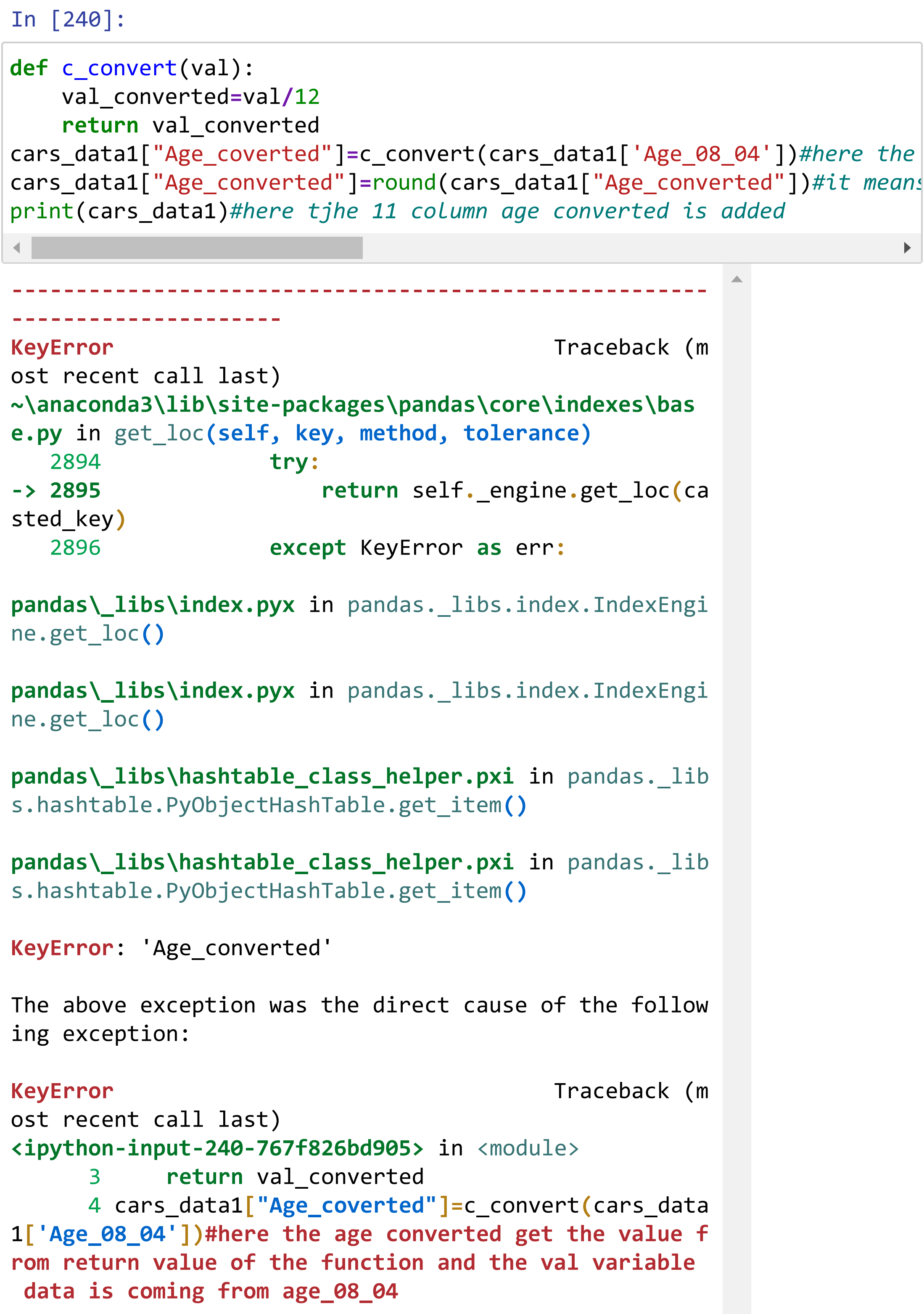
In [239]:

```
#Exam

ple:function
#Converting the age variable from months to years by defining a functio
#the converted values will be stored in a new column,'Age_cconverted'
#hence inserting a new column
cars_data1.insert(11,'Age_converted',0)#here 11 means the position of A
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-239-5533251ec9c4> in <module>
      1 #Exam
      2 
----> 3 ple:function
      4 #Converting the age variable from months to years by defining a function
      5 #the converted values will be stored in a new column,'Age_cconverted'

NameError: name 'function' is not defined
```



In [240]:

```
def c_convert(val):
    val_converted=val/12
    return val_converted
cars_data1["Age_coverted"]=c_convert(cars_data1['Age_08_04'])#here the
cars_data1["Age_converted"]=round(cars_data1["Age_converted"])#it means
print(cars_data1)#here tjhe 11 column age converted is added
```

```
---------------------------------------------------------------------
---------------------
KeyError                                  Traceback (m
ost recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\bas
e.py in get_loc(self, key, method, tolerance)
   2894             try:
-> 2895                 return self._engine.get_loc(ca
sted_key)
   2896             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngi
ne.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngi
ne.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._lib
s.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._lib
s.hashtable.PyObjectHashTable.get_item()

KeyError: 'Age_converted'

The above exception was the direct cause of the follow
ing exception:

KeyError                                  Traceback (m
ost recent call last)
<ipython-input-240-767f826bd905> in <module>
      3     return val_converted
      4 cars_data1["Age_coverted"]=c_convert(cars_data
1['Age_08_04'])#here the age converted get the value f
rom return value of the function and the val variable
 data is coming from age_08_04
```

```
----> 5 cars_data1["Age_converted"]=round(cars_data1[
"Age_converted"])#it means age converted column"s data
would be round upto the one decimal point
      6 print(cars_data1)#here tjhe 11 column age conv
erted is added

~\anaconda3\lib\site-packages\pandas\core\frame.py in
__getitem__(self, key)
   2900             if self.columns.nlevels > 1:
   2901                 return self._getitem_multileve
l(key)
-> 2902             indexer = self.columns.get_loc(key
)
   2903             if is_integer(indexer):
   2904                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\bas
e.py in get_loc(self, key, method, tolerance)
   2895                 return self._engine.get_loc(ca
sted_key)
   2896             except KeyError as err:
-> 2897                 raise KeyError(key) from err
   2898 
   2899         if tolerance is not None:

KeyError: 'Age_converted'
```

Function with multiple inputs and outputs

```
#Function with multiple inputs and a single ouput
1)Funtion in python takes ,ultiple inputs objects return only one
object as output
2)however lists,tuples or dictionaries can be used to return multiple
output as required
```

In [241]:

```
cars_data1.insert(12,"KM_per_month",0)#here the 12 column is created n
```

In [242]:

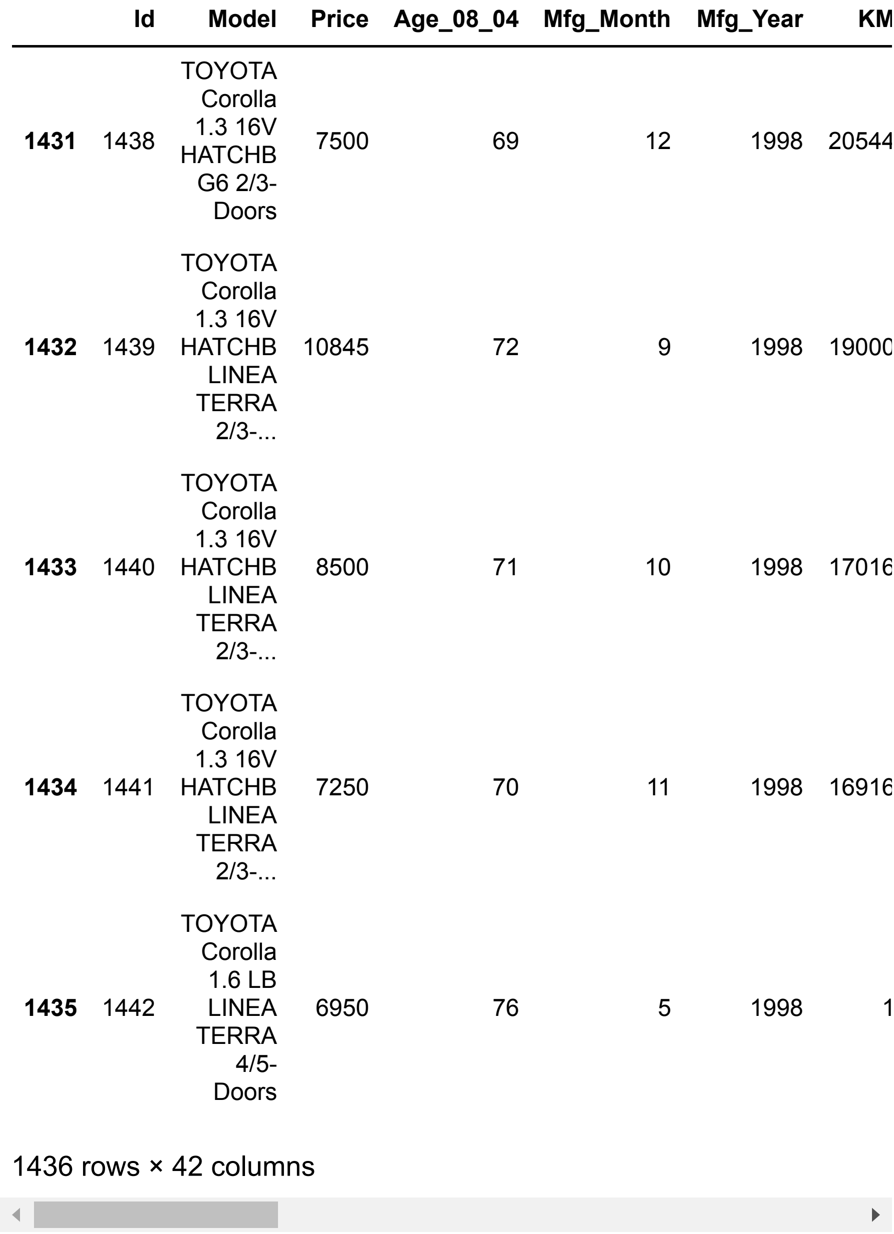
```
#A multiple input multiple output function c_convert
#the function taken in two input
#the output is returned in the form of a list
def c_convert (val1,val2):
    val_converted=val1/12
    ratio=val2/val1
    return [val_converted,ratio]#it will return it form of list
```

In [243]:

```
#here Age and km columns of the data set are input to the function func
#the outputs are assigned to 'Age_converted' and 'Km_per_month'
cars_data1["Age_converted"],cars_data1["Km_per_month"]=  c_convert(cars
cars_data1
```

Out[243]:

	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002	46986
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002	72937
2	3	? TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002	41711
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	7	2002	48000
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002	38500
...	...	...	...	...	...	...	...



	Id	Model	Price	Age_08_04	Mfg_Month	Mfg_Year	KM
1431	1438	TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors	7500	69	12	1998	20544
1432	1439	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	10845	72	9	1998	19000
1433	1440	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	8500	71	10	1998	17016
1434	1441	TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...	7250	70	11	1998	16916
1435	1442	TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors	6950	76	5	1998	1

1436 rows × 42 columns

In [ ]:

In [244]:

```python
import pandas as pd
cars_data=pd.read_csv('ToyotaCorolla.csv',index_col=0,na_values=["??",'
print(cars_data)
```

```
                                                  Model
Price  Age_08_04  \
Id
1         TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors
13500         23
2         TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors
13750         23
3        ?TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors
13950         24
4         TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors
14950         26
5           TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors
13750         30
...                                                 ...
...        ...
1438         TOYOTA Corolla 1.3 16V HATCHB G6 2/3-Doors
7500         69
1439  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...
10845         72
1440  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...
8500         71
1441  TOYOTA Corolla 1.3 16V HATCHB LINEA TERRA 2/3-...
7250         70
1442        TOYOTA Corolla 1.6 LB LINEA TERRA 4/5-Doors
6950         76

      Mfg_Month  Mfg_Year     KM Fuel_Type   HP  Met_Col
or  Automatic  ...  \
Id
...
1            10      2002  46986    Diesel   90
1          0  ...
2            10      2002  72937    Diesel   90
1          0  ...
3             9      2002  41711    Diesel   90
1          0  ...
4             7      2002  48000    Diesel   90
0          0  ...
5             3      2002  38500    Diesel   90
0          0  ...
```

```
...         ...       ...    ...       ...  ...
...        ...  ...
1438         12       1998  20544    Petrol   86
1          0  ...
1439          9       1998  19000    Petrol   86
0          0  ...
1440         10       1998  17016    Petrol   86
0          0  ...
1441         11       1998  16916    Petrol   86
1          0  ...
1442          5       1998      1    Petrol  110
0          0  ...

      Central_Lock  Powered_Windows  Power_Steering  Rad
io  Mistlamps  \
Id
1                1                1               1
0          0
2                1                0               1
0          0
3                0                0               1
0          0
4                0                0               1
0          0
5                1                1               1
0          1
...            ...              ...             ...
...        ...
1438             1                1               1
0          1
1439             0                0               1
0          0
1440             0                0               1
0          0
1441             0                0               0
0          0
1442             0                0               1
0          0

      Sport_Model  Backseat_Divider  Metallic_Rim  Radio
_cassette  Tow_Bar
Id
1               0                 1             0
0        0
2               0                 1             0
0        0
```

```
3               0                 1             0
0        0
4               0                 1             0
0        0
5               0                 1             0
0        0
...            ...               ...           ...
...      ...
1438            1                 1             0
0        0
1439            1                 1             0
0        0
1440            0                 1             0
0        0
1441            0                 1             0
0        0
1442            0                 0             0
0        0

[1436 rows x 36 columns]
```

In [245]:

```
#crating a copy of data
cars_data2=cars_data.copy()#the data change in cars_data2 will not refl
```

In [246]:

```
#here we are considiring only one categorial value hat is fuel type
pd.crosstab(index=cars_data2['Fuel_Type'],columns="count",dropna=True)#
#count gives the count of the category fuel_type
#dopna =true will drop the null values
```

Out[246]:

col_0	count
Fuel_Type	
CNG	17
Diesel	155
Petrol	1264

In [247]:

```
# Frequency tables
```

```
1)to compute a simple cross-tabulation of one or two (or more)
factors
2)by default computes  frquency table of factors
```

In [248]:

```
#here we are cosidiring two categorical variable hence we are using two
#to look the frequency distribution of gearbox types with respect to di
pd.crosstab(index=cars_data2['Automatic'],columns=cars_data2['Fuel_Type
#here  automatic corresponds to column and fuel typpe considered as row
#here 0 represent manual gear box and 1 represent the Automatic gear bo
#here manual gear box has 16 cng,155 diesel,1185 petrol
#here automatic ear box has 1 cng ,0 diesel 79 petrol
```

Out[248]:

Fuel_Type	CNG	Diesel	Petrol
Automatic			
0	16	155	1185
1	1	0	79

In [249]:

```
pd.crosstab(index=cars_data2['Automatic'],columns=cars_data2['Fuel_Type
#by normalise is equal to true means we are converting number into prop
```

Out[249]:

Fuel_Type	CNG	Diesel	Petrol
Automatic			
0	0.011142	0.107939	0.825209
1	0.000696	0.000000	0.055014

Two way marginal probablity

In [250]:

```
#pandas.crosstab()
```

marginal probablity is the probablity of the occurance of the single event

In [251]:

```
import pandas as pd
pd.crosstab(index=cars_data2['Automatic'],columns=cars_data2['Fuel_Type
#we did normalize is equal to true because we want every thing in propo
#by setting margins is equal to true we get the sum of all row and colu
```

Out[251]:

Fuel_Type	CNG	Diesel	Petrol	All
Automatic				
0	0.011142	0.107939	0.825209	0.94429
1	0.000696	0.000000	0.055014	0.05571
All	0.011838	0.107939	0.880223	1.00000

Two way conditional probablity

1)conditional probablity is the probablity of an event (A),given that another event (B) has already occured
2)Given the type of gear box,probablity of different fuel type

In [252]:

```
pd.crosstab(index=cars_data2['Automatic'],columns=cars_data2['Fuel_Type
#by declaring normal=index we get conditional probablity basicall we do
#here we get the probablity of all cng
```

Out[252]:

Fuel_Type	CNG	Diesel	Petrol
Automatic			
0	0.011799	0.114307	0.873894
1	0.012500	0.000000	0.987500
All	0.011838	0.107939	0.880223

In [253]:

```
#to get the sum of 0
pd.crosstab(index=cars_data2['Automatic'],columns=cars_data2['Fuel_Type
```

Out[253]:

Fuel_Type	CNG	Diesel	Petrol	All
Automatic				
0	0.941176	1.0	0.9375	0.94429
1	0.058824	0.0	0.0625	0.05571

Correlation

```
1)Correlation:The strength of association between two variable
2)Visual representation of correlation Scatter plots
```

In [254]:

```
#Dataframe.corr(self,method='person')
```

```
1)To compute PAIRWISE correlation of columns N/A null values
```

```
2)Excluding the categorical variable to find the person's correlation
3)person is used to check the strength of two numerical values
```

In [255]:

```
numerical_data=cars_data2.select_dtypes(exclude=[object])#here we are s
```

In [256]:

```
#checking the no of variable available under numerical_data
print(numerical_data.shape)
```

```
(1436, 34)
```

In [258]:

```
#for creating matrix
corr_matrix=numerical_data.corr()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: