Today's Objective

- Hackerarh prb
- Data Science Libraries
  - Numpy
  - Pandas
  - Matplot library

In [2]:
```
# e-maz-in
#     Input:LLRDDR
#     out put:0 -2
s=input()
x=0
y=0
for var in s:
    if var=="L":
        x=x-1
    elif var=="R":
        x=x+1
    elif var == "U":
        y=y+1
    elif var == "D":
        y=y-1
print(x,y)

```

```
LLRDDR
0 -2
```

Data Science

```
- Numpy (Numarical Python)
```

- NumPy (Numerical Python) is a linear algebra library in Python. It is a very important library on which almost every data science or machine learning Python packages such as SciPy (Scientific Python), Mat−plotlib (plotting library), Scikit-learn, etc depends on to a reasonable extent.
  - NumPy is very useful for performing mathematical and logical operations on Arrays. It provides an abundance of useful features for operations on n-arrays and matrices in Python.

In [1]:
```
import numpy as np
#print(s1.size*s1.itemsize)
#print(sys.getsizeof(s)*len(s))
```

In [5]:
```
# Normal List
import sys

# print(list(range(1001)))
s1=range(1000)
print(sys.getsizeof(s1)*len(s1))
```

```
48000
```

In [7]:
```
#In Numpy
n=np.arange(1000)
print(n.size*n.itemsize)
```

```
4000
```

In [8]:
```
np.__version__
```

Out[8]:
```
'1.16.4'
```

In [4]:
```
1 print(dir(np))
```

```
['ALLOW_THREADS', 'AxisError', 'BUFSIZE', 'CLIP', 'ComplexWarning', 'DataSourc
e', 'ERR_CALL', 'ERR_DEFAULT', 'ERR_IGNORE', 'ERR_LOG', 'ERR_PRINT', 'ERR_RAIS
E', 'ERR_WARN', 'FLOATING_POINT_SUPPORT', 'FPE_DIVIDEBYZERO', 'FPE_INVALID', 'F
PE_OVERFLOW', 'FPE_UNDERFLOW', 'False_', 'Inf', 'Infinity', 'MAXDIMS', 'MAY_SHA
RE_BOUNDS', 'MAY_SHARE_EXACT', 'MachAr', 'ModuleDeprecationWarning', 'NAN', 'NI
NF', 'NZERO', 'NaN', 'PINF', 'PZERO', 'RAISE', 'RankWarning', 'SHIFT_DIVIDEBYZE
RO', 'SHIFT_INVALID', 'SHIFT_OVERFLOW', 'SHIFT_UNDERFLOW', 'ScalarType', 'Teste
r', 'TooHardError', 'True_', 'UFUNC_BUFSIZE_DEFAULT', 'UFUNC_PYVALS_NAME', 'Vis
ibleDeprecationWarning', 'WRAP', '_NoValue', '_UFUNC_API', '__NUMPY_SETUP__',
'__all__', '__builtins__', '__cached__', '__config__', '__doc__', '__file__',
'__git_revision__', '__loader__', '__mkl_version__', '__name__', '__package__',
'__path__', '__spec__', '__version__', '_add_newdoc_ufunc', '_arg', '_distribut
or_init', '_globals', '_mat', '_mklinit', '_pytesttester', 'abs', 'absolute',
'absolute_import', 'add', 'add_docstring', 'add_newdoc', 'add_newdoc_ufunc', 'a
len', 'all', 'allclose', 'alltrue', 'amax', 'amin', 'angle', 'any', 'append',
'apply_along_axis', 'apply_over_axes', 'arange', 'arccos', 'arccosh', 'arcsin',
'arcsinh', 'arctan', 'arctan2', 'arctanh', 'argmax', 'argmin', 'argpartition',
'argsort', 'argwhere', 'around', 'array', 'array2string', 'array_equal', 'array
_equiv', 'array_repr', 'array_split', 'array_str', 'asanyarray', 'asarray', 'as
array_chkfinite', 'ascontiguousarray', 'asfarray', 'asfortranarray', 'asmatri
x', 'asscalar', 'atleast_1d', 'atleast_2d', 'atleast_3d', 'average', 'bartlet
t', 'base_repr', 'binary_repr', 'bincount', 'bitwise_and', 'bitwise_not', 'bitw
ise_or', 'bitwise_xor', 'blackman', 'block', 'bmat', 'bool', 'bool8', 'bool_',
'broadcast', 'broadcast_arrays', 'broadcast_to', 'busday_count', 'busday_offse
t', 'busdaycalendar', 'byte', 'byte_bounds', 'bytes0', 'bytes_', 'c_', 'can_cas
t', 'cast', 'cbrt', 'cdouble', 'ceil', 'cfloat', 'char', 'character', 'chararra
y', 'choose', 'clip', 'clongdouble', 'clongfloat', 'column_stack', 'common_typ
e', 'compare_chararrays', 'compat', 'complex', 'complex128', 'complex64', 'comp
lex_', 'complexfloating', 'compress', 'concatenate', 'conj', 'conjugate', 'conv
olve', 'copy', 'copysign', 'copyto', 'core', 'corrcoef', 'correlate', 'cos', 'c
osh', 'count_nonzero', 'cov', 'cross', 'csingle', 'ctypeslib', 'cumprod', 'cump
roduct', 'cumsum', 'datetime64', 'datetime_as_string', 'datetime_data', 'deg2ra
d', 'degrees', 'delete', 'deprecate', 'deprecate_with_doc', 'diag', 'diag_indic
es', 'diag_indices_from', 'diagflat', 'diagonal', 'diff', 'digitize', 'disp',
'divide', 'division', 'divmod', 'dot', 'double', 'dsplit', 'dstack', 'dtype',
'e', 'ediff1d', 'einsum', 'einsum_path', 'emath', 'empty', 'empty_like', 'equa
l', 'errstate', 'euler_gamma', 'exp', 'exp2', 'expand_dims', 'expm1', 'extrac
t', 'eye', 'fabs', 'fastCopyAndTranspose', 'fft', 'fill_diagonal', 'find_common
_type', 'finfo', 'fix', 'flatiter', 'flatnonzero', 'flexible', 'flip', 'flipl
r', 'flipud', 'float', 'float16', 'float32', 'float64', 'float_', 'float_powe
r', 'floating', 'floor', 'floor_divide', 'fmax', 'fmin', 'fmod', 'format_float_
positional', 'format_float_scientific', 'format_parser', 'frexp', 'frombuffer',
'fromfile', 'fromfunction', 'fromiter', 'frompyfunc', 'fromregex', 'fromstrin
g', 'full', 'full_like', 'fv', 'gcd', 'generic', 'genfromtxt', 'geomspace', 'ge
t_array_wrap', 'get_include', 'get_printoptions', 'getbufsize', 'geterr', 'gete
rrcall', 'geterrobj', 'gradient', 'greater', 'greater_equal', 'half', 'hammin
g', 'hanning', 'heaviside', 'histogram', 'histogram2d', 'histogram_bin_edges',
'histogramdd', 'hsplit', 'hstack', 'hypot', 'i0', 'identity', 'iinfo', 'imag',
'in1d', 'index_exp', 'indices', 'inexact', 'inf', 'info', 'infty', 'inner', 'in
sert', 'int', 'int0', 'int16', 'int32', 'int64', 'int8', 'int_', 'int_asbuffe
r', 'intc', 'integer', 'interp', 'intersect1d', 'intp', 'invert', 'ipmt', 'ir
r', 'is_busday', 'isclose', 'iscomplex', 'iscomplexobj', 'isfinite', 'isfortra
n', 'isin', 'isinf', 'isnan', 'isnat', 'isneginf', 'isposinf', 'isreal', 'isrea
lobj', 'isscalar', 'issctype', 'issubclass_', 'issubdtype', 'issubsctype', 'ite
rable', 'ix_', 'kaiser', 'kron', 'lcm', 'ldexp', 'left_shift', 'less', 'less_eq
```

```
ual', 'lexsort', 'lib', 'linalg', 'linspace', 'little_endian', 'load', 'loads',
'loadtxt', 'log', 'log10', 'log1p', 'log2', 'logaddexp', 'logaddexp2', 'logical
_and', 'logical_not', 'logical_or', 'logical_xor', 'logspace', 'long', 'longcom
plex', 'longdouble', 'longfloat', 'longlong', 'lookfor', 'ma', 'mafromtxt', 'ma
sk_indices', 'mat', 'math', 'matmul', 'matrix', 'matrixlib', 'max', 'maximum',
'maximum_sctype', 'may_share_memory', 'mean', 'median', 'memmap', 'meshgrid',
'mgrid', 'min', 'min_scalar_type', 'minimum', 'mintypecode', 'mirr', 'mod', 'mo
df', 'moveaxis', 'msort', 'multiply', 'nan', 'nan_to_num', 'nanargmax', 'nanarg
min', 'nancumprod', 'nancumsum', 'nanmax', 'nanmean', 'nanmedian', 'nanmin', 'n
anpercentile', 'nanprod', 'nanquantile', 'nanstd', 'nansum', 'nanvar', 'nbyte
s', 'ndarray', 'ndenumerate', 'ndfromtxt', 'ndim', 'ndindex', 'nditer', 'negati
ve', 'nested_iters', 'newaxis', 'nextafter', 'nonzero', 'not_equal', 'nper', 'n
pv', 'numarray', 'number', 'obj2sctype', 'object', 'object0', 'object_', 'ogri
d', 'oldnumeric', 'ones', 'ones_like', 'outer', 'packbits', 'pad', 'partition',
'percentile', 'pi', 'piecewise', 'place', 'pmt', 'poly', 'poly1d', 'polyadd',
'polyder', 'polydiv', 'polyfit', 'polyint', 'polymul', 'polynomial', 'polysub',
'polyval', 'positive', 'power', 'ppmt', 'print_function', 'printoptions', 'pro
d', 'product', 'promote_types', 'ptp', 'put', 'put_along_axis', 'putmask', 'p
v', 'quantile', 'r_', 'rad2deg', 'radians', 'random', 'rank', 'rate', 'ravel',
'ravel_multi_index', 'real', 'real_if_close', 'rec', 'recarray', 'recfromcsv',
'recfromtxt', 'reciprocal', 'record', 'remainder', 'repeat', 'require', 'reshap
e', 'resize', 'result_type', 'right_shift', 'rint', 'roll', 'rollaxis', 'root
s', 'rot90', 'round', 'round_', 'row_stack', 's_', 'safe_eval', 'save', 'savetx
t', 'savez', 'savez_compressed', 'sctype2char', 'sctypeDict', 'sctypeNA', 'scty
pes', 'searchsorted', 'select', 'set_numeric_ops', 'set_printoptions', 'set_str
ing_function', 'setbufsize', 'setdiff1d', 'seterr', 'seterrcall', 'seterrobj',
'setxor1d', 'shape', 'shares_memory', 'short', 'show_config', 'sign', 'signbi
t', 'signedinteger', 'sin', 'sinc', 'single', 'singlecomplex', 'sinh', 'size',
'sometrue', 'sort', 'sort_complex', 'source', 'spacing', 'split', 'sqrt', 'squa
re', 'squeeze', 'stack', 'std', 'str', 'str0', 'str_', 'string_', 'subtract',
'sum', 'swapaxes', 'sys', 'take', 'take_along_axis', 'tan', 'tanh', 'tensordo
t', 'test', 'testing', 'tile', 'timedelta64', 'trace', 'tracemalloc_domain', 't
ranspose', 'trapz', 'tri', 'tril', 'tril_indices', 'tril_indices_from', 'trim_z
eros', 'triu', 'triu_indices', 'triu_indices_from', 'true_divide', 'trunc', 'ty
peDict', 'typeNA', 'typecodes', 'typename', 'ubyte', 'ufunc', 'uint', 'uint0',
'uint16', 'uint32', 'uint64', 'uint8', 'uintc', 'uintp', 'ulonglong', 'unicod
e', 'unicode_', 'union1d', 'unique', 'unpackbits', 'unravel_index', 'unsignedin
teger', 'unwrap', 'ushort', 'vander', 'var', 'vdot', 'vectorize', 'version', 'v
oid', 'void0', 'vsplit', 'vstack', 'warnings', 'where', 'who', 'zeros', 'zeros_
like']
```

Numpy Vs List

- Numpy is better then List following main three reasons
  - It is very Fast
  - It is Less Memory
  - It is Convineant

In [13]:
```
# Single Dimentional array

import numpy as np

a=np.array([1,2,3,4,5])
print("Single Dimentional array",a)
li=[1,2,3,4,5]
print("this the list",li)
print(len(a))
print(len(li))
```

```
Single Dimentional array [1 2 3 4 5]
this the list [1, 2, 3, 4, 5]
5
5
```

In [15]:
```
#Multiply with 10 normal list
li*10
```

Out[15]:
```
[1,
 2,
 3,
 4,
 5,
 1,
 2,
 3,
 4,
 5,
 1,
 2,
 3,
 4,
 5,
 1,
 2,
 3,
 4,
```

In [17]:
```
# By using Numpy
a*10
```

Out[17]:
```
array([10, 20, 30, 40, 50])
```

In [19]:
```
#add the each with 3
print("adding 3 in every numpy list  elemnt",a+3)
print("adding 3 in every list ele",li+3)
```

```
adding 3 in every numpy list  elemnt [4 5 6 7 8]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-2245abac1cac> in <module>
      1 #add the each with 3
      2 print("adding 3 in every numpy list  elemnt",a+3)
----> 3 print("adding 3 in every list ele",li+3)

TypeError: can only concatenate list (not "int") to list
```

In [20]:
```
print("squring each and  every numpy list ele",a**2)
```

```
squring each and  every numpy list ele [ 1  4  9 16 25]
```

In [21]:
```
print("subtract each and  every numpy list ele",a-3)
```

```
subtract each and  every numpy list ele [-2 -1  0  1  2]
```

In [22]:
```
li
```

Out[22]:
```
[1, 2, 3, 4, 5]
```

In [26]:
```
#Normal list Squring
for elem in range(len(li)):
    print(li[elem]**2,end=" ")
```

```
1 4 9 16 25
```

In [31]:
```
#Single dimem[] random gen..
single=np.random.randint(10,size=5)
single
```

Out[31]:
```
array([6, 5, 6, 7, 0])
```

In [37]:
```
#Two dimentional arry
t=np.array([(1,2,3,4,5),(6,7,8,9)])
t
```

Out[37]:
```
array([(1, 2, 3, 4, 5), (6, 7, 8, 9)], dtype=object)
```

In [40]:
```
twoD=np.random.randint(10,size=(3,2))
print("two dimentional array")
twoD
```

```
two dimentional array
```

Out[40]:
```
array([[2, 5],
       [1, 9],
       [6, 6]])
```

In [43]:
```
1 twoD[1][1]#Indexing
```

Out[43]:
```
9
```

In [54]:
```
1 twoD[1]=[3,7]
2
```

In [55]:
```
1 twoD
```

Out[55]:
```
array([[2, 5],
       [3, 7],
       [6, 6]])
```

In [58]:
```
1 twoD[0:2]#slicing
```

Out[58]:
```
array([[2, 5],
       [3, 7]])
```

In [59]:
```
1 twoD.T#trasnspose
```

Out[59]:
```
array([[2, 3, 6],
       [5, 7, 6]])
```

In [63]:
```
1 twoD//4
```

Out[63]:
```
array([[0, 1],
       [0, 1],
       [1, 1]], dtype=int32)
```

In [66]:
```
#Multi Dimentional array
m=np.random.randint(100,size=(5,4,5))
print(m)
```

```
[[[22 76 54 59 91]
  [55 53  2 81 13]
  [28 77 54 43 40]
  [ 6  0 91 23 53]]

 [[27 33 55 87 20]
  [22 90  3 18 82]
  [29 74 42 41 20]
  [ 6 52 71  9 46]]

 [[94 75 29 76 50]
  [68 15 49 35 45]
  [63 20 42 42 63]
  [93 64 78 59  3]]

 [[ 9 52  4 49  9]
  [35  1 92 88 90]
  [78 69 98 10 39]
  [14 88  2 20 89]]

 [[13 54 30 15  9]
  [17 76  9  5 24]
  [75 35 51 87 18]
  [44 31 13 21  5]]]
```

In [69]:
```
m[:3]
```

Out[69]:
```
array([[[22, 76, 54, 59, 91],
        [55, 53,  2, 81, 13],
        [28, 77, 54, 43, 40],
        [ 6,  0, 91, 23, 53]],

       [[27, 33, 55, 87, 20],
        [22, 90,  3, 18, 82],
        [29, 74, 42, 41, 20],
        [ 6, 52, 71,  9, 46]],

       [[94, 75, 29, 76, 50],
        [68, 15, 49, 35, 45],
        [63, 20, 42, 42, 63],
        [93, 64, 78, 59,  3]]])
```

In [70]:
```
1 print(dir(m))
```

```
['T', '__abs__', '__add__', '__and__', '__array__', '__array_finalize__', '__ar
ray_function__', '__array_interface__', '__array_prepare__', '__array_priority_
_', '__array_struct__', '__array_ufunc__', '__array_wrap__', '__bool__', '__cla
ss__', '__complex__', '__contains__', '__copy__', '__deepcopy__', '__delattr_
_', '__delitem__', '__dir__', '__divmod__', '__doc__', '__eq__', '__float__',
'__floordiv__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__g
t__', '__hash__', '__iadd__', '__iand__', '__ifloordiv__', '__ilshift__', '__im
atmul__', '__imod__', '__imul__', '__index__', '__init__', '__init_subclass__',
'__int__', '__invert__', '__ior__', '__ipow__', '__irshift__', '__isub__', '__i
ter__', '__itruediv__', '__ixor__', '__le__', '__len__', '__lshift__', '__lt_
_', '__matmul__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__or_
_', '__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__reduce__',
'__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__', '__rmatmul__', '__
rmod__', '__rmul__', '__ror__', '__rpow__', '__rrshift__', '__rshift__', '__rsu
b__', '__rtruediv__', '__rxor__', '__setattr__', '__setitem__', '__setstate__',
'__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv__', '__xor_
_', 'all', 'any', 'argmax', 'argmin', 'argpartition', 'argsort', 'astype', 'bas
e', 'byteswap', 'choose', 'clip', 'compress', 'conj', 'conjugate', 'copy', 'cty
pes', 'cumprod', 'cumsum', 'data', 'diagonal', 'dot', 'dtype', 'dump', 'dumps',
'fill', 'flags', 'flat', 'flatten', 'getfield', 'imag', 'item', 'itemset', 'ite
msize', 'max', 'mean', 'min', 'nbytes', 'ndim', 'newbyteorder', 'nonzero', 'par
tition', 'prod', 'ptp', 'put', 'ravel', 'real', 'repeat', 'reshape', 'resize',
'round', 'searchsorted', 'setfield', 'setflags', 'shape', 'size', 'sort', 'sque
eze', 'std', 'strides', 'sum', 'swapaxes', 'take', 'tobytes', 'tofile', 'tolis
t', 'tostring', 'trace', 'transpose', 'var', 'view']
```

In [71]:
```
1 m.shape
```

Out[71]:
```
(5, 4, 5)
```

In [72]:
```
1 m.size
```

Out[72]:
```
100
```

In [73]:
```
1 twoD
```

Out[73]:
```
array([[2, 5],
       [3, 7],
       [6, 6]])
```

In [77]:
```
1 twoD.resize?
```

In [90]:
```
1 # a = np.array([[0, 1], [2, 3]], order='C')
2 # a.resize((2, 1))
3 # a
4 twoD.resize((1,2,3))
```

In [88]:
```
1 twoD
```

Out[88]:
```
array([[[2, 5, 3],
        [7, 6, 6]]])
```

In [91]:
```
1 twoD.ndim
```

Out[91]:
```
3
```

In [101]:
```
1 z=np.zeros(3)
2 z
3 o=np.ones(4)
4 o*4.5
```

Out[101]:
```
array([4.5, 4.5, 4.5, 4.5])
```

In [127]:
```
1 c=np.arange(0.1,10,0.2)
2 print(sum(c))
3 c
```

```
250.00000000000003
```

Out[127]:
```
array([0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9, 2.1, 2.3, 2.5,
       2.7, 2.9, 3.1, 3.3, 3.5, 3.7, 3.9, 4.1, 4.3, 4.5, 4.7, 4.9, 5.1,
       5.3, 5.5, 5.7, 5.9, 6.1, 6.3, 6.5, 6.7, 6.9, 7.1, 7.3, 7.5, 7.7,
       7.9, 8.1, 8.3, 8.5, 8.7, 8.9, 9.1, 9.3, 9.5, 9.7, 9.9])
```

In [128]:
```
1 np.sqrt(c)
```

Out[128]:
```
array([0.31622777, 0.54772256, 0.70710678, 0.83666003, 0.9486833 ,
       1.04880885, 1.14017543, 1.22474487, 1.30384048, 1.37840488,
       1.44913767, 1.51657509, 1.58113883, 1.64316767, 1.70293864,
       1.76068169, 1.81659021, 1.87082869, 1.92353841, 1.97484177,
       2.02484567, 2.07364414, 2.12132034, 2.16794834, 2.21359436,
       2.25831796, 2.30217289, 2.34520788, 2.38746728, 2.42899156,
       2.46981781, 2.50998008, 2.54950976, 2.58843582, 2.62678511,
       2.66458252, 2.70185122, 2.73861279, 2.77488739, 2.81069386,
       2.84604989, 2.88097206, 2.91547595, 2.94957624, 2.98328678,
       3.01662063, 3.04959014, 3.082207  , 3.1144823 , 3.14642654])
```

In [ ]:
```
1 task
2 # 2 4 6
3 # 6 8 14
```

In [137]:
```
1 se=np.array([(2,4),(6,8)])
2 se
```

Out[137]:
```
array([[2, 4],
       [6, 8]])
```

In [138]:
```
1 r1=se[0][0]+se[0][1]
2 r1
```

Out[138]:
```
6
```

In [139]:
```
1 r2=se[1][0]+se[1][1]
2 r2
```

Out[139]:
```
14
```

In [136]:
```
1 se
```

Out[136]:
```
array([array([[2, 4],
       [6, 8]]), 6, 14], dtype=object)
```

In [157]:
```
1 newMatx=np.array([[r1,r2],[8,5]])
```

In [158]:
```
1 newMatx
```

Out[158]:
```
array([[ 6, 14],
       [ 8,  5]])
```

In [159]:
```
1 np.concatenate((se,newMatx))
```

Out[159]:
```
array([[ 2,  4],
       [ 6,  8],
       [ 6, 14],
       [ 8,  5]])
```

Pandas

- Usecases
  - Data Trasformation
  - Data visuvalization
  - Data Cleaning
- Natations
  - Series Data Set
  - DataFrames

In [160]:
```
import pandas as pd

print(dir(pd))
```

```
['Categorical', 'CategoricalDtype', 'CategoricalIndex', 'DataFrame', 'DateOffse
t', 'DatetimeIndex', 'DatetimeTZDtype', 'ExcelFile', 'ExcelWriter', 'Float64Ind
ex', 'Grouper', 'HDFStore', 'Index', 'IndexSlice', 'Int16Dtype', 'Int32Dtype',
'Int64Dtype', 'Int64Index', 'Int8Dtype', 'Interval', 'IntervalDtype', 'Interval
Index', 'MultiIndex', 'NaT', 'Panel', 'Period', 'PeriodDtype', 'PeriodIndex',
'RangeIndex', 'Series', 'SparseArray', 'SparseDataFrame', 'SparseDtype', 'Spars
eSeries', 'TimeGrouper', 'Timedelta', 'TimedeltaIndex', 'Timestamp', 'UInt16Dty
pe', 'UInt32Dtype', 'UInt64Dtype', 'UInt64Index', 'UInt8Dtype', '__builtins__',
'__cached__', '__doc__', '__docformat__', '__file__', '__git_version__', '__loa
der__', '__name__', '__package__', '__path__', '__spec__', '__version__', '_has
htable', '_lib', '_libs', '_np_version_under1p13', '_np_version_under1p14', '_n
p_version_under1p15', '_np_version_under1p16', '_np_version_under1p17', '_tsli
b', '_version', 'api', 'array', 'arrays', 'bdate_range', 'compat', 'concat', 'c
ore', 'crosstab', 'cut', 'date_range', 'datetime', 'describe_option', 'errors',
'eval', 'factorize', 'get_dummies', 'get_option', 'infer_freq', 'interval_rang
e', 'io', 'isna', 'isnull', 'lreshape', 'melt', 'merge', 'merge_asof', 'merge_o
rdered', 'notna', 'notnull', 'np', 'offsets', 'option_context', 'options', 'pan
das', 'period_range', 'pivot', 'pivot_table', 'plotting', 'qcut', 'read_clipboa
rd', 'read_csv', 'read_excel', 'read_feather', 'read_fwf', 'read_gbq', 'read_hd
f', 'read_html', 'read_json', 'read_msgpack', 'read_parquet', 'read_pickle', 'r
ead_sas', 'read_sql', 'read_sql_query', 'read_sql_table', 'read_stata', 'read_t
able', 'reset_option', 'set_eng_float_format', 'set_option', 'show_versions',
'test', 'testing', 'timedelta_range', 'to_datetime', 'to_msgpack', 'to_numeri
c', 'to_pickle', 'to_timedelta', 'tseries', 'unique', 'util', 'value_counts',
'wide_to_long']
```

In [161]:
```
# Series Data Notations
fmid={'psa':24,'dsp':12,'emf':9,'m4':30,"psd":28,"ds":2}
type(fmid)
```

Out[161]:
```
dict
```

In [162]:
```
#Series
fmid=pd.Series(fmid)
fmid
```

Out[162]:
```
psa    24
dsp    12
emf     9
m4     30
psd    28
ds      2
dtype: int64
```

In [163]:
```
smid={'psa':30,'dsp':29,'emf':30,'m4':4,"psd":18,"ds":30}
smid=pd.Series(smid)
smid
```

Out[163]:
```
psa    30
dsp    29
emf    30
m4      4
psd    18
ds     30
dtype: int64
```

In [170]:
```
df={"First Mid marks ":fmid,"Second Mid Marks ":smid}
df=pd.DataFrame(df)
print(df)
df.columns[1]

```

```
     First Mid marks   Second Mid Marks
psa                24                 30
dsp                12                 29
emf                 9                 30
m4                 30                  4
psd                28                 18
ds                  2                 30
```

Out[170]:
```
'Second Mid Marks '
```

In [171]:
```
df.values
```

Out[171]:
```
array([[24, 30],
       [12, 29],
       [ 9, 30],
       [30,  4],
       [28, 18],
       [ 2, 30]], dtype=int64)
```

In [172]:
```
df.values[0][0]
```

Out[172]:
```
24
```

In [173]:
```
#sum of fmid + smid marks
df.values[0][0]+df.values[0][1]
```

Out[173]:
```
54
```

In [ ]:
```

```