Numpy

In [1]:
```
import numpy as np
```

In [2]:
```
a = np.array([1,2,3,4,5,6,7,8,9,0])
a
```

Out[2]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9, 0])
```

In [3]:
```
a.reshape(2,5)
```

Out[3]:
```
array([[1, 2, 3, 4, 5],
       [6, 7, 8, 9, 0]])
```

In [4]:
```
a.reshape(2,5,1)
```

Out[4]:
```
array([[[1],
        [2],
        [3],
        [4],
        [5]],

       [[6],
        [7],
        [8],
        [9],
        [0]]])
```

In [5]:
```
# concatenation
a1 = np.array([1,2,3,4])
a2 = np.array([6,7,8,9,10])
```

In [6]:
```
np.concatenate((a1,a2))
```

Out[6]:
```
array([ 1,  2,  3,  4,  6,  7,  8,  9, 10])
```

In [7]:
```
a3 = np.array([[1,2,3],[5,6,7]])
```

In [8]:
```
np.concatenate((a1,a3))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-908de27af4f1> in <module>
----> 1 np.concatenate((a1,a3))

ValueError: all the input arrays must have same number of dimensions
```

In [9]:
```
a4 = np.array([[4,6,7],[9,2,3]])
a4
```

Out[9]:
```
array([[4, 6, 7],
       [9, 2, 3]])
```

In [10]:
```
a3
```

Out[10]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [11]:
```
np.concatenate((a3,a4))
```

Out[11]:
```
array([[1, 2, 3],
       [5, 6, 7],
       [4, 6, 7],
       [9, 2, 3]])
```

In [12]:
```
np.concatenate((a3,a4),axis=1)
```

Out[12]:
```
array([[1, 2, 3, 4, 6, 7],
       [5, 6, 7, 9, 2, 3]])
```

In [13]:
```
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']
```

In [14]:
```
1 a3
```

Out[14]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [15]:
```
1 np.min(a3) # minimum element
```

Out[15]:
```
1
```

In [16]:
```
1 np.max(a3)
```

Out[16]:
```
7
```

In [17]:
```
1 np.argmin(a3) # index value minimum element
```

Out[17]:
```
0
```

In [18]:
```
1 np.argmax(a3)
```

Out[18]:
```
5
```

In [19]:
```
1 np.mean(a3)
```

Out[19]:
```
4.0
```

In [20]:
```
1 np.median(a3)
```

Out[20]:
```
4.0
```

In [21]:
```
1 np.average(a3)
```

Out[21]:
```
4.0
```

In [22]:
```
1 np.var(a3) # variance
```

Out[22]:
```
4.666666666666667
```

In [23]:
```
1 np.std(a3) # stanard deviation
```

Out[23]:
```
2.160246899469287
```

In [24]:
```
1 np.sum(a3)
```

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

In [25]:
```
1 a3
```

Out[25]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [26]:
```
1 np.cumsum(a3) # cumulative sum
```

Out[26]:
```
array([ 1,  3,  6, 11, 17, 24], dtype=int32)
```

In [27]:
```
1 a3
```

Out[27]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [28]:
```
1 np.min(a3)
```

Out[28]:
```
1
```

In [29]:
```
np.min(a3,axis=1)  # 1-- rows
```

Out[29]:
```
array([1, 5])
```

In [30]:
```
# axis :  1---rows
#          0--- columns
```

In [31]:
```
np.min(a3,axis=0)
```

Out[31]:
```
array([1, 2, 3])
```

In [32]:
```
np.max(a3,axis=1)  # 1-- rows
```

Out[32]:
```
array([3, 7])
```

In [33]:
```
np.argmax(a3,axis=0)
```

Out[33]:
```
array([1, 1, 1], dtype=int64)
```

In [34]:
```
np.argmax(a3,0)
```

Out[34]:
```
array([1, 1, 1], dtype=int64)
```

In [35]:
```
np.min(a3[1])
```

Out[35]:
```
5
```

In [36]:
```
a3[1]
```

Out[36]:
```
array([5, 6, 7])
```

In [37]:
```
a3
```

Out[37]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [38]:
```
# stacking - arranging elements in proper order

# horizontal stacking
# vertical stacking
```

In [39]:
```
a3
```

Out[39]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [40]:
```
1 np.hstack(a3)
```

Out[40]:
```
array([1, 2, 3, 5, 6, 7])
```

In [41]:
```
1 np.vstack(a3)
```

Out[41]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [42]:
```
1 np.sqrt(a3)
```

Out[42]:
```
array([[1.        , 1.41421356, 1.73205081],
       [2.23606798, 2.44948974, 2.64575131]])
```

In [43]:
```
1 np.exp(a3)
```

Out[43]:
```
array([[   2.71828183,    7.3890561 ,   20.08553692],
       [ 148.4131591 ,  403.42879349, 1096.63315843]])
```

In [44]:
```
1 np.log(a3)
```

Out[44]:
```
array([[0.        , 0.69314718, 1.09861229],
       [1.60943791, 1.79175947, 1.94591015]])
```

In [45]:
```
1 np.log2(a3)
```

Out[45]:
```
array([[0.        , 1.        , 1.5849625 ],
       [2.32192809, 2.5849625 , 2.80735492]])
```

In [46]:
```
1 np.log10(a3)
```

Out[46]:
```
array([[0.        , 0.30103   , 0.47712125],
       [0.69897   , 0.77815125, 0.84509804]])
```

In [47]:
```
1 np.remainder(a3,4)
```

Out[47]:
```
array([[1, 2, 3],
       [1, 2, 3]], dtype=int32)
```

In [48]:
```
1 a3
```

Out[48]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [49]:
```
1 np.divide(a3,4)
```

Out[49]:
```
array([[0.25, 0.5 , 0.75],
       [1.25, 1.5 , 1.75]])
```

In [50]:
```
np.power(a3,2)
```

Out[50]:
```
array([[ 1,  4,  9],
       [25, 36, 49]], dtype=int32)
```

In [51]:
```
np.multiply(a3,5)
```

Out[51]:
```
array([[ 5, 10, 15],
       [25, 30, 35]])
```

In [52]:
```
s = np.array([1+2j,5+6j])
s
```

Out[52]:
```
array([1.+2.j, 5.+6.j])
```

In [53]:
```
print(np.real(s))
print(np.imag(s))
```

```
[1. 5.]
[2. 6.]
```

In [54]:
```
l = [1,2,3,4]
l*3
```

Out[54]:
```
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
```

In [55]:
```
a3
```

Out[55]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [56]:
```
a3*3
```

Out[56]:
```
array([[ 3,  6,  9],
       [15, 18, 21]])
```

Random methods

In [57]:
```
np.random.random() # it returns random value between 0 and 1
```

Out[57]:
```
0.9859560714514989
```

In [58]:
```
np.random.random(10)
```

Out[58]:
```
array([0.77028641, 0.82217241, 0.36857966, 0.99396245, 0.84443301,
       0.46910941, 0.39970311, 0.38324724, 0.19560851, 0.75056124])
```

In [59]:
```
1 np.random.random((2,5,2))
```

Out[59]:
```
array([[[0.00575808, 0.95422415],
        [0.86996526, 0.67996042],
        [0.30147184, 0.56228083],
        [0.20602066, 0.93946113],
        [0.86006609, 0.40849877]],

       [[0.92378569, 0.74747216],
        [0.07632888, 0.40783029],
        [0.82166701, 0.03081718],
        [0.66702618, 0.13866014],
        [0.90270504, 0.36990381]]])
```

In [60]:
```
1 np.random.randint(5) # range of 5
```

Out[60]:
```
0
```

In [61]:
```
1 np.random.randint(5,20)
```

Out[61]:
```
8
```

Filtering

In [62]:
```
1 a = np.array([34,67,56,38,67,90,100,58])
```

In [63]:
```
1 a
```

Out[63]:
```
array([ 34,  67,  56,  38,  67,  90, 100,  58])
```

In [64]:
```
1 a > 50
```

Out[64]:
```
array([False,  True,  True, False,  True,  True,  True,  True])
```

In [65]:
```
1 a[a>50]
```

Out[65]:
```
array([ 67,  56,  67,  90, 100,  58])
```

In [66]:
```
1 l = [34,67,56,38,67,90,100,58]
2 l
```

Out[66]:
```
[34, 67, 56, 38, 67, 90, 100, 58]
```

In [67]:
```
l>50
#-- 67,56,67,90,100,58
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-67-b290700b24be> in <module>
----> 1 l>50
      2 #-- 67,56,67,90,100,58

TypeError: '>' not supported between instances of 'list' and 'int'
```

In [68]:
```
x = np.arange(25,100)
```

In [69]:
```
x
```

Out[69]:
```
array([25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
       42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
       59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75,
       76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
       93, 94, 95, 96, 97, 98, 99])
```

In [70]:
```
x[x%2==0]
```

Out[70]:
```
array([26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58,
       60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92,
       94, 96, 98])
```

In [71]:
```
x[(x>30)&(x<60)]

```

Out[71]:
```
array([31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
       48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59])
```

In [72]:
```
x[((x>30)&(x<60))&(x%2==0)]

```

Out[72]:
```
array([32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58])
```

In [73]:
```

# | ---or
# !----not
```

Pandas

- import/exportthe datsets
- analysize and manipulate the data

Differnt types of files (https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html)

Types:

- Series
    - list,tuple,numpy array
- DataFrame
    - list,tuple,dict,numpy array

- Datatypes -- float,int,string/object

In [74]:
```
import pandas as pd
```

In [ ]:
```
# pip install pandas
# pip install numpy
```

In [ ]:
```
pip install pandas
```

In [75]:
```
z = pd.Series([1,2,3,4])
z
```

Out[75]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

In [76]:
```
pd.Series((89,0.8,8))
```

Out[76]:
```
0    89.0
1     0.8
2     8.0
dtype: float64
```

In [77]:
```
pd.Series(np.array([1,2,3,4]))
```

Out[77]:
```
0    1
1    2
2    3
3    4
dtype: int32
```

In [78]:
```
a3
```

Out[78]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [79]:
```
1 pd.Series(a3)
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
<ipython-input-79-04efc429a814> in <module>
----> 1 pd.Series(a3)

~\Anaconda3\lib\site-packages\pandas\core\series.py in __init__(self, data, ind
ex, dtype, name, copy, fastpath)
    260             else:
    261                 data = sanitize_array(data, index, dtype, copy,
--> 262                                       raise_cast_failure=True)
    263
    264                 data = SingleBlockManager(data, index, fastpath=True)

~\Anaconda3\lib\site-packages\pandas\core\internals\construction.py in sanitize
_array(data, index, dtype, copy, raise_cast_failure)
    656     elif subarr.ndim > 1:
    657         if isinstance(data, np.ndarray):
--> 658             raise Exception('Data must be 1-dimensional')
    659         else:
    660             subarr = com.asarray_tuplesafe(data, dtype=dtype)

Exception: Data must be 1-dimensional
```

In [80]:
```
1 v = pd.Series([2,"mvgr",90],index=["a","b","c"])
2 v
```

Out[80]:
```
a       2
b    mvgr
c      90
dtype: object
```

In [81]:
```
1 v.index
```

Out[81]:
```
Index(['a', 'b', 'c'], dtype='object')
```

In [82]:
```
1 v = pd.Series([2,"mvgr",90],index=["a","b","c"],columns=["ap"])
2 v
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-82-1efc20e0ee9e> in <module>
----> 1 v = pd.Series([2,"mvgr",90],index=["a","b","c"],columns=["ap"])
      2 v

TypeError: __init__() got an unexpected keyword argument 'columns'
```

In [83]:
```
1 pd.Series({"a":[67,89,90],"b":90})
```

Out[83]:
```
a    [67, 89, 90]
b              90
dtype: object
```

In [84]:
```
1 a3
```

Out[84]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [85]:
```
1 d = pd.DataFrame(a3,index = ["a",5],columns=["mvgr","raghu","gates"])
2 d
```

Out[85]:

	mvgr	raghu	gates
a	1	2	3
5	5	6	7

In [86]:
```
1 d
```

Out[86]:

	mvgr	raghu	gates
a	1	2	3
5	5	6	7

In [87]:
```
1 pd.DataFrame({"a":[67,89,90],"b":90})
```

Out[87]:

	a	b
0	67	90
1	89	90
2	90	90

In [88]:
```
1 pd.Series({"a":[67,89,90],"b":90})
```

Out[88]:
```
a    [67, 89, 90]
b              90
dtype: object
```

In [89]:
```
1 pd.DataFrame([[1,2,3],[5,6]],index=("a","b"),columns = ("x","y","z"))
```

Out[89]:

	x	y	z
a	1	2	3.0
b	5	6	NaN

In [90]:
```
#pd.DataFrame(((2,5,6),(5,8,90)),index=(1,2),columns=("a","h","c"))
```

In [92]:
```
#pd.DataFrame(((2,5,6),(5,8,90)),index=(1,2),columns=("a","h","c"))
```

In [93]:
```
d = {"Name":["apssdc","workshop"],"date":[12,6],"year":[2012,2013]}
```

In [97]:
```
d1 = pd.DataFrame(d,index=["a","b"])
```

In [98]:
```
d1
```

Out[98]:

	Name	date	year
a	apssdc	12	2012
b	workshop	6	2013

```

```

In [99]:
```
d1.columns
```

Out[99]:
```
Index(['Name', 'date', 'year'], dtype='object')
```

In [100]:
```
d1.index
```

Out[100]:
```
Index(['a', 'b'], dtype='object')
```

In [101]:
```
d1.shape
```

Out[101]:
```
(2, 3)
```

In [102]:
```
# Accessing the elements
```

In [103]:
```
d1["Name"]
```

Out[103]:
```
a      apssdc
b    workshop
Name: Name, dtype: object
```

In [106]:
```
d1["year"]
```

Out[106]:
```
a    2012
b    2013
Name: year, dtype: int64
```

In [109]:
```
1 d1.year
```

Out[109]:
```
a    2012
b    2013
Name: year, dtype: int64
```

In [108]:
```
1 d1[["Name","year"]]
```

Out[108]:

	Name	year
a	apssdc	2012
b	workshop	2013

In [112]:
```
1 d1["Name"]["b"]
```

Out[112]:
```
'workshop'
```

In [113]:
```
1 d1["Name"][1]
```

Out[113]:
```
'workshop'
```

In [114]:
```
1 d1["Name"]
```

Out[114]:
```
a      apssdc
b    workshop
Name: Name, dtype: object
```

In [115]:
```
1 d1
```

Out[115]:

	Name	date	year
a	apssdc	12	2012
b	workshop	6	2013

In [117]:
```
1 d1[1:]
```

Out[117]:

	Name	date	year
b	workshop	6	2013

In [121]:
```
1 d1[-1:]
```

Out[121]:

	Name	date	year
b	workshop	6	2013

In [ ]:
```
1
```