Get your hands dirty 1

1. Install virtualenv

- Open CMD window, and go to your working path
- Type: pip freeze ; To list all installed python packages
- Type: pip install virtualenv ; To install virtualenv
- Type: virtualenv your-project-title, e.g., virtualenv lab1
- Go inside your-project-title or lab1 folder
- Type: Script\activate ; To activate virtualenv
- Type: Script\deactivate ; To deactivate virtualenv

reference: shorturl.at/dlxX0

2. Install visual studio code

reference: shorturl.at/vGJ27

3. Data collection

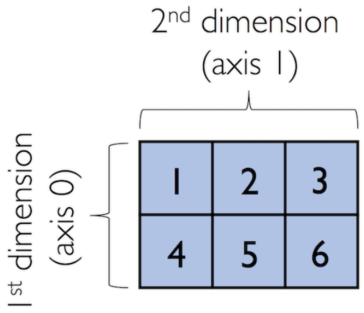
Data collection	Can be ordered	Can be changeable	Can be duplicated
List	yes	yes	yes
Tuple	yes	no	yes
Set	no	no	no
Dictionary	no	yes	no

4. Numpy

4.1 N-dimensional arrays

```
>>> import numpy as np
>>> l = [[1,2,3],[4,5,6]]
>>> type(l)
<class ‘list'>
>>> a2d = np.array(l)
>>> a2d
array([[1, 2, 3],
       [4, 5, 6]])
>>> type(a2d)
<class 'numpy.ndarray'>
>>> a2d.size
6
```

Get your hands dirty 1




```
>>> a2d.ndim
2

>>> a2d.shape
(2, 3)

>>> np.array([1,2]).shape #the one-dimensional array
(2,)
```

4.2 Ones

```
>>> np.ones((2,2))
array([[1., 1.],
       [1., 1.]])

>>> x = np.ones((2,2))
>>> x.dtype
dtype('float64')
```

4.3 Zeros

```
>>> np.zeros((3,3))
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

4.4 arange

```
np.arange(4,10)   # a half-open interval
>>> array([4, 5, 6, 7, 8, 9])

np.arange(4)
>>> array([0, 1, 2, 3])

np.arange(4,10, 2)
>>> array([4, 6, 8])
```

Get your hands dirty 1

4.5 linspace

```
np.linspace(0,1,5)
>>> array([0.  , 0.25, 0.5 , 0.75, 1.  ])
```

4.6 Identity matrix

```
>>> np.eye(5)
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

4.7 Diagonal matrix

```
>>> np.diag((2,3,4))
array([[2, 0, 0],
       [0, 3, 0],
       [0, 0, 4]])
```

4.8 Array indexing

```
One dimension (1D)
>>> a1d = np.array([1,2,3])
>>> a1d
array([1, 2, 3])

>>> a1d[0]
1

>>> a1d[:2]
array([1, 2])

Two dimensions (2D)
>>> a2d
array([[1, 2, 3],
       [4, 5, 6]])

>>> a2d[0,0]
1

>>> a2d[1,2]
6
```

```
>>> a2d[-1,-1]
6

>>> a2d[-1,-2]
5

>>> a2d[:,0]
array([1, 4])

>>> a2d[:, :2]
array([[1, 2], [4, 5]])
```

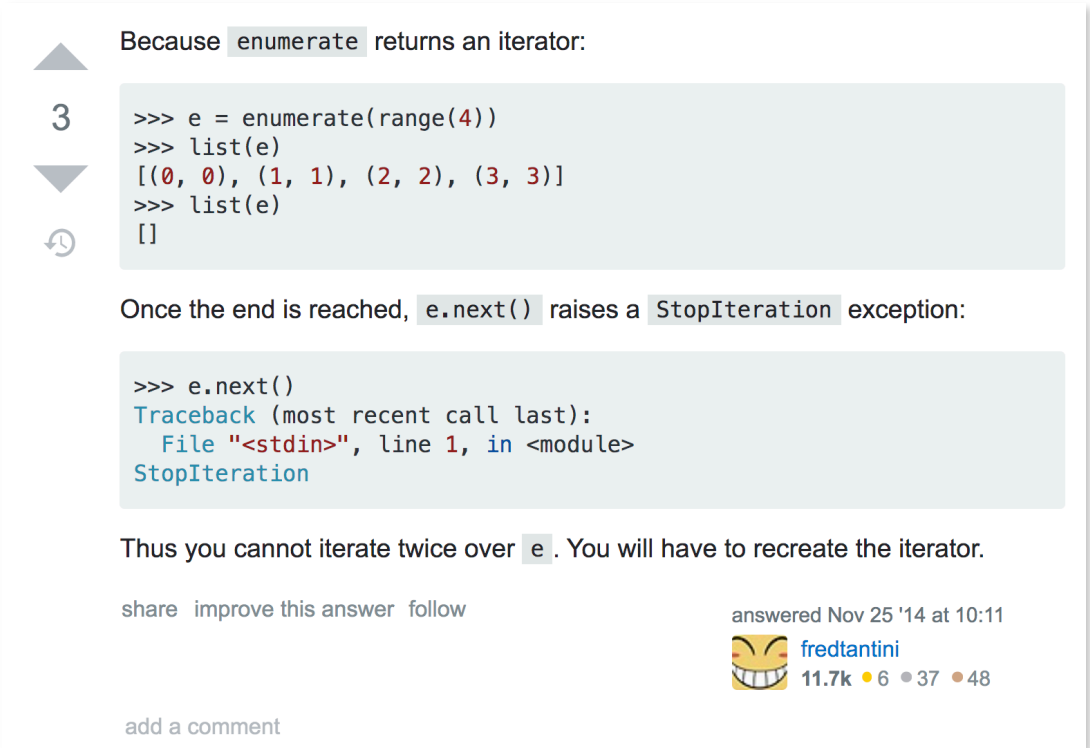
4.9 Enumerate

```
>>> l
[[1, 2, 3], [4, 5, 6]]

>>> for row_idx, row_val in enumerate(l):
...     for col_idx, col_val in enumerate(row_val):
...             l[row_idx][col_idx] += 1
...

>>> l
[[2, 3, 4], [5, 6, 7]]
```

Be careful when to use enumerate function



3

Because `enumerate` returns an iterator:

```
>>> e = enumerate(range(4))
>>> list(e)
[(0, 0), (1, 1), (2, 2), (3, 3)]
>>> list(e)
[]
```

Once the end is reached, `e.next()` raises a `StopIteration` exception:

```
>>> e.next()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

Thus you cannot iterate twice over `e`. You will have to recreate the iterator.

share improve this answer follow

answered Nov 25 '14 at 10:11

fredtantini

11.7k • 6 • 37 • 48

add a comment

Get your hands dirty 1

4.10 List comprehension style

```
>>> l2 = [[cell + 1 for cell in row] for row in l]
>>> l2
[[3, 4, 5], [6, 7, 8]]
```

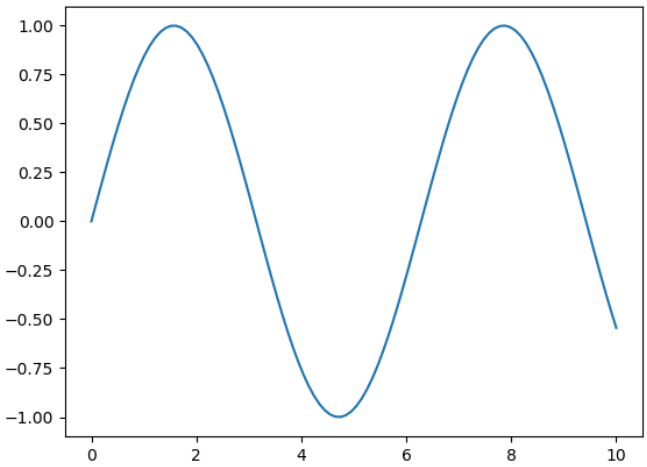
4.11 Reshaping arrays

```
>>> a2d_reshape = a2d.reshape(3,2)
>>> a2d_reshape
array([[1, 2],
       [3, 4],
       [5, 6]])
```

5. Matplotlib

```
>>> x = np.linspace(0,10,100)
>>> plt.plot(x, np.sin(x))
[<matplotlib.lines.Line2D object at 0x120c492d0>]

>>> plt.show()
```



6. Matplotlib Animation

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

x = np.linspace(0,10,100)
y = np.sin(x)
```

```
fig, ax = plt.subplots()
line1, = ax.plot(x, y, color = "r")

def update(num, x, y, line1):
    line1.set_data(x[:num], y[:num])
    return [line1]

ani = animation.FuncAnimation(fig, update, len(x), fargs=[x, y, line1],
                interval=100, blit=True)
ax.set_xlabel('X')
ax.set_ylabel('Y')
plt.show()
```

7. Time measurement

```
#import time
from timeit import default_timer as timer
start = timer()
# do something here
end = timer()
print(end - start)
```

Example [https://www.geeksforgeeks.org/vectorization-in-python/]

```
# Dot product
import time
import numpy
import array

# 8 bytes size int
a = array.array('q')
for i in range(100000):
        a.append(i);

b = array.array('q')
for i in range(100000, 200000):
        b.append(i)

# classic dot product of vectors implementation
tic = time.process_time()
dot = 0.0;

for i in range(len(a)):
        dot += a[i] * b[i]
```

Get your hands dirty 1

```
toc = time.process_time()
print("dot_product = "+ str(dot));
print("Computation time = " + str(1000*(toc - tic )) + "ms")

n_tic = time.process_time()
n_dot_product = numpy.dot(a, b)
n_toc = time.process_time()

print("\nn_dot_product = "+str(n_dot_product))
print("Computation time = "+str(1000*(n_toc - n_tic ))+"ms")
```

Output:

```
dot_product = 833323333350000.0
Computation time = 35.59449199999999 ms
```

```
n_dot_product = 833323333350000
Computation time = 0.1559900000000225 ms
```

8. Miscellaneous

Help function

```
>>  help(np.ones)
Help on function ones in module numpy:
ones(shape, dtype=None, order='C')
    Return a new array of given shape and type, filled with ones.
    Parameters
    ----------
    shape : int or sequence of ints
        Shape of the new array, e.g., ``(2, 3)`` or ``2``.
    dtype : data-type, optional
                The desired data-type for the array, e.g., `numpy.int8`.
          Default is `numpy.float64`.
    order : {'C', 'F'}, optional, default: C
        Whether to store multi-dimensional data in row-major
        (C-style) or column-major (Fortran-style) order in
        memory.
    Returns
    -------
    out : ndarray
        Array of ones with the given shape, dtype, and order.
```

Get your hands dirty 1

How many functions of numpy?

```
>>> dir(np)
['ALLOW_THREADS', 'AxisError', 'BUFSIZE', 'CLIP', 'ComplexWarning', 'DataSource', 'ERR_CALL', 'ERR_DEFAU
LT', 'ERR_IGNORE', 'ERR_LOG', 'ERR_PRINT', 'ERR_RAISE', 'ERR_WARN', 'FLOATING_POINT_SUPPORT', 'FPE_DIVID
EBYZERO', 'FPE_INVALID', 'FPE_OVERFLOW', 'FPE_UNDERFLOW', 'False_', 'Inf', 'Infinity', 'MAXDIMS', 'MAY_S
HARE_BOUNDS', 'MAY_SHARE_EXACT', 'MachAr', 'ModuleDeprecationWarning', 'NAN', 'NINF', 'NZERO', 'NaN', 'P
INF', 'PZERO', 'RAISE', 'RankWarning', 'SHIFT_DIVIDEBYZERO', 'SHIFT_INVALID', 'SHIFT_OVERFLOW', 'SHIFT_U
NDERFLOW', 'ScalarType', 'Tester', 'TooHardError', 'True_', 'UFUNC_BUFSIZE_DEFAULT', 'UFUNC_PYVALS_NAME'
, 'VisibleDeprecationWarning', 'WRAP', '_NoValue', '_UFUNC_API', '__NUMPY_SETUP__', '__all__', '__builti
ns__', '__cached__', '__config__', '__doc__', '__file__', '__git_revision__', '__loader__', '__mkl_versi
on__', '__name__', '__package__', '__path__', '__spec__', '__version__', '_add_newdoc_ufunc', '_distribu
tor_init', '_globals', '_mat', '_pytesttester', 'abs', 'absolute', 'absolute_import', 'add', 'add_docstr
ing', 'add_newdoc', 'add_newdoc_ufunc', 'alen', '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', 'asanya
rray', 'asarray', 'asarray_chkfinite', 'ascontiguousarray', 'asfarray', 'asfortranarray', 'asmatrix', 'a
sscalar', 'atleast_1d', 'atleast_2d', 'atleast_3d', 'average', 'bartlett', 'base_repr', 'binary_repr', '
bincount', 'bitwise_and', 'bitwise_not', 'bitwise_or', 'bitwise_xor', 'blackman', 'block', 'bmat', 'bool
', 'bool8', 'bool_', 'broadcast', 'broadcast_arrays', 'broadcast_to', 'busday_count', 'busday_offset', '
busdaycalendar', 'byte', 'byte_bounds', 'bytes0', 'bytes_', 'c_', 'can_cast', 'cast', 'cbrt', 'cdouble',
 'ceil', 'cfloat', 'char', 'character', 'chararray', 'choose', 'clip', 'clongdouble', 'clongfloat', 'col
umn_stack', 'common_type', 'compare_chararrays', 'compat', 'complex', 'complex128', 'complex256', 'compl
ex64', 'complex_', 'complexfloating', 'compress', 'concatenate', 'conj', 'conjugate', 'convolve', 'copy'
, 'copysign', 'copyto', 'core', 'corrcoef', 'correlate', 'cos', 'cosh', 'count_nonzero', 'cov', 'cross',
 'csingle', 'ctypeslib', 'cumprod', 'cumproduct', 'cumsum', 'datetime64', 'datetime_as_string', 'datetim
e_data', 'deg2rad', 'degrees', 'delete', 'deprecate', 'deprecate_with_doc', 'diag', 'diag_indices', 'dia
g_indices_from', 'diagflat', 'diagonal', 'diff', 'digitize', 'disp', 'divide', 'division', 'divmod', 'do
t', 'double', 'dsplit', 'dstack', 'dtype', 'e', 'ediff1d', 'einsum', 'einsum_path', 'emath', 'empty', 'e
mpty_like', 'equal', 'errstate', 'euler_gamma', 'exp', 'exp2', 'expand_dims', 'expm1', 'extract', 'eye',
 'fabs', 'fastCopyAndTranspose', 'fft', 'fill_diagonal', 'find_common_type', 'finfo', 'fix', 'flatiter',
 'flatnonzero', 'flexible', 'flip', 'fliplr', 'flipud', 'float', 'float128', 'float16', 'float32', 'floa
t64', 'float_', 'float_power', 'floating', 'floor', 'floor_divide', 'fmax', 'fmin', 'fmod', 'format_floa
t_positional', 'format_float_scientific', 'format_parser', 'frexp', 'frombuffer', 'fromfile', 'fromfunct
ion', 'fromiter', 'frompyfunc', 'fromregex', 'fromstring', 'full', 'full_like', 'fv', 'gcd', 'generic',
'genfromtxt', 'geomspace', 'get_array_wrap', 'get_include', 'get_printoptions', 'getbufsize', 'geterr',
'geterrcall', 'geterrobj', 'gradient', 'greater', 'greater_equal', 'half', 'hamming', 'hanning', 'heavis
ide', 'histogram', 'histogram2d', 'histogram_bin_edges', 'histogramdd', 'hsplit', 'hstack', 'hypot', 'i0
', 'identity', 'iinfo', 'imag', 'in1d', 'index_exp', 'indices', 'inexact', 'inf', 'info', 'infty', 'inne
r', 'insert', 'int', 'int0', 'int16', 'int32', 'int64', 'int8', 'int_', 'int_asbuffer', 'intc', 'integer
', 'interp', 'intersect1d', 'intp', 'invert', 'ipmt', 'irr', 'is_busday', 'isclose', 'iscomplex', 'iscom
plexobj', 'isfinite', 'isfortran', 'isin', 'isinf', 'isnan', 'isnat', 'isneginf', 'isposinf', 'isreal',
'isrealobj', 'isscalar', 'issctype', 'issubclass_', 'issubdtype', 'issubsctype', 'iterable', 'ix_', 'kai
ser', 'kron', 'lcm', 'ldexp', 'left_shift', 'less', 'less_equal', 'lexsort', 'lib', 'linalg', 'linspace'
```