lec3_step2

November 30, 2022

[1]:

```
## Python basics for novice data scientists, supported by Wagatsuma Lab@Kyutech
#
# The MIT License (MIT): Copyright (c) 2020 Hiroaki Wagatsuma and Wagatsuma␣
 ↪Lab@Kyutech
#
# Permission is hereby granted, free of charge, to any person obtaining a copy␣
 ↪of this software and associated documentation files (the "Software"), to␣
 ↪deal in the Software without restriction, including without limitation the␣
 ↪rights to use, copy, modify, merge, publish, distribute, sublicense, and/or␣
 ↪sell copies of the Software, and to permit persons to whom the Software is␣
 ↪furnished to do so, subject to the following conditions:
# The above copyright notice and this permission notice shall be included in␣
 ↪all copies or substantial portions of the Software.
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR␣
 ↪IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,␣
 ↪FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE␣
 ↪AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER␣
 ↪LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING␣
 ↪FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS␣
 ↪IN THE SOFTWARE. */
#
# # @Time    : 2022-8-20
# # @Author  : Hiroaki Wagatsuma
# # @Site    : https://github.com/hirowgit/2A1_python_intermediate_course
# # @IDE     : Python 3.9.13 (main, Aug  7 2022, 01:33:23) [Clang 13.1.6␣
 ↪(clang-1316.0.21.2.5)] on darwin
# # @File    : lec3_step2.py
```

[8]:

```
import numpy as np
prFill=np.array([90, 60, 50, 50, 50, 90, 40, 30, 80, 40, 20])
prFill=prFill/100
print(prFill)
```

```
[0.9 0.6 0.5 0.5 0.5 0.9 0.4 0.3 0.8 0.4 0.2]
```

[17]:

```
a=[10,20,30]
print(a)
```

```
for d in a:
    print(d/10)
for i in range(len(a)):
    print(a[i]/10)
```

```
[10, 20, 30]
1.0
2.0
3.0
1.0
2.0
3.0
```

[139]:

```
import numpy as np
#prFill=[90    60    50    50    50    90    40    30    80    40  20 ]/100;
prFill=np.array([90, 60, 50, 50, 50, 90, 40, 30, 80, 40, 20])
prFill=prFill/100
print(prFill)
```

```
[0.9 0.6 0.5 0.5 0.5 0.9 0.4 0.3 0.8 0.4 0.2]
```

[18]:

```
# fillLine=boolean(ones(1,size(prFill,2)));
#  1
fillLine=np.full(len(prFill),True)
print(fillLine)
```

```
[ True  True  True  True  True  True  True  True  True  True  True]
```

[3]:

```
# fillLine=boolean(ones(1,size(prFill,2)));
#  1
NofD=len(prFill)
fillLine=np.full(NofD,True)
print(fillLine)
```

```
[ True  True  True  True  True  True  True  True  True  True  True]
```

[88]:

```
# fillLine=boolean(ones(1,size(prFill,2)));
#  1    :       x
fillLine=np.full(( 1,len(prFill)),True)
print(fillLine)
```

```
[[ True  True  True  True  True  True  True  True  True  True  True]]
```

[23]:

```
#
fillLine2=np.empty(len(prFill), dtype = bool)
fillLine2[:]=True  #
print(fillLine2)
```

```
[ True  True  True  True  True  True  True  True  True  True  True]
```

[41]:

```
a=np.zeros((2,3))
print(a)
print(len(a))
print(a.size)
print(a.ndim)
print(a.shape)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
2
6
2
(2, 3)
```

[19]:

```
b=np.argwhere(prFill>0.8)
print(b)
print(b.shape)
```

```
[[0]
 [5]]
(2, 1)
```

[143]:

```
prFill
```

[143]:

```
array([0.9, 0.6, 0.5, 0.5, 0.5, 0.9, 0.4, 0.3, 0.8, 0.4, 0.2])
```

[152]:

```
c=np.where(prFill>0.8)
print(c)
print(np.shape(c))
```

```
(array([0, 5]),)
(1, 2)
```

[47]:

```
np.argwhere((prFill>0.8) & fillLine)
```

[47]:

```
array([[0, 0],
       [0, 5]])
```

[156]:

```
fillLine[5]=False
np.where((prFill>0.8) & fillLine)
```

[156]:

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

[158]:

```
fillLine
```

```
[158]: array([ True,  True,  True,  True,  True, False,  True,  True,  True,
                True,  True])
```

```
[159]: prFill
```

```
[159]: array([0.9, 0.6, 0.5, 0.5, 0.5, 0.9, 0.4, 0.3, 0.8, 0.4, 0.2])
```

```
[54]: prFill[(prFill>0.8) ]
```

```
[54]: array([0.9, 0.9])
```

```
[55]: np.where((prFill>0.8) & (prFill>=0.8))
```

```
[55]: (array([0, 5]),)
```

```
[90]: b=np.where(np.logical_and(prFill>0.8,fillLine))
      print(b)
      print(len(b))
```

```
(array([0, 5]),)
1
```

```
[69]: b2=np.unique(b)
      print(b2)
```

```
[0 5]
```

```
[71]: b2[0]
```

```
[71]: 0
```

```
[91]: print(fillLine[3:-1])
```

```
[ True  True  True  True  True  True  True]
```

```
[135]: i=3
       np.where((prFill[i+1:-1]>0.8) & fillLine[i+1:-1])
```

```
[135]: (array([1]),)
```

```
[137]: i=3
       remF=0.5
       IDrem=np.where((prFill[i+1:-1]>remF) & fillLine[i+1:-1])
       print(IDrem)
```

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

[134]:

```
a=np.empty(0)
print(a)
a.append([0,1])
print(a)
np.append(a,np.array([4])
print(a)
np.append(a,np.array([1,2,3])
```

```
  File "<ipython-input-134-26a9b3048070>", line 6
    print(a)
    ^
SyntaxError: invalid syntax
```

[20]:

```
a=[]
print(a)
a.append([0,1])
print(a)
a.append([1,2,3])
print(a)
```

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

[129]:

```
gg=[[1,2],[0,1,2],[5]]
print(gg)
print(gg[0])
gg.append([1,2,1,1,3])
print(gg)
```

```
[[1, 2], [0, 1, 2], [5]]
[1, 2]
[[1, 2], [0, 1, 2], [5], [1, 2, 1, 1, 3]]
```

[11]:

```
np.array([[1,2,3],[4,5,6]])
```

[11]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

[12]:

```
np.array([[1,2,3],[4,5]])
```

```
/var/folders/mg/w5t8lkhc8xj79f001s7kzpfh0000gp/T/ipykernel_46824/4191096310.py:1
: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
```

```
'dtype=object' when creating the ndarray.
  np.array([[1,2,3],[4,5]])
```

```
[12]: array([list([1, 2, 3]), list([4, 5])], dtype=object)
```

```
[16]: np.array([['a','a'],['b','b']])
```

```
[16]: array([['a', 'a'],
             ['b', 'b']], dtype='<U1')
```

```
[17]: np.array([['a','a'],['b','b','c']])
```

```
/var/folders/mg/w5t8lkhc8xj79f001s7kzpfh0000gp/T/ipykernel_46824/3154860643.py:1
: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
  np.array([['a','a'],['b','b','c']])
```

```
[17]: array([list(['a', 'a']), list(['b', 'b', 'c'])], dtype=object)
```

```
[28]: d=[[1,2,3],[4,5]]
      print(d)
```

```
[[1, 2, 3], [4, 5]]
```

```
[29]: d[0]
```

```
[29]: [1, 2, 3]
```

```
[30]: d[0][2]
```

```
[30]: 3
```

```
[18]: [['a','a'],['b','b','c']]
```

```
[18]: [['a', 'a'], ['b', 'b', 'c']]
```

```
[23]: c=['aa','bbc']
      print(c)
```

```
['aa', 'bbc']
```

```
[25]: c[0]
```

```
[25]: 'aa'
```

```
[26]: c[1][2]
```

[26]:

```
'c'
```

[40]:

```
import sympy as sym
```

[85]:

```
A =sym.MatrixSymbol('a',2,2)
b =sym.MatrixSymbol('b',2,1)
A.as_explicit()
```

[85]:

$$\begin{bmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \end{bmatrix}$$

[87]:

```
b.as_explicit()
```

[87]:

$$\begin{bmatrix} b_{0,0} \\ b_{1,0} \end{bmatrix}$$

[86]:

```
y=A*b
y.as_explicit()
```

[86]:

$$\begin{bmatrix} a_{0,0}b_{0,0} + a_{0,1}b_{1,0} \\ a_{1,0}b_{0,0} + a_{1,1}b_{1,0} \end{bmatrix}$$

[91]:

```
sym.init_printing
```

[91]:

```
<function sympy.interactive.printing.init_printing(pretty_print=True,
order=None, use_unicode=None, use_latex=None, wrap_line=None, num_columns=None,
no_global=False, ip=None, euler=False, forecolor=None, backcolor='Transparent',
fontsize='10pt', latex_mode='plain', print_builtin=True, str_printer=None,
pretty_printer=None, latex_printer=None, scale=1.0, **settings)>
```

[65]:

```
A=sym.Matrix([[1, 2], [3, 4]])
A
```

[65]:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

[64]:

```
b=sym.Matrix([[5], [6]])
b
```

[64]:

$$\begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

[66]:

```
A*b
```

[66]:

$$\begin{bmatrix} 17 \\ 39 \end{bmatrix}$$

[74]:

```
sym.MatrixSymbol('a',2,2)*sym.MatrixSymbol('b',2,1)[1]
```

[74]:

$b_{1,0}a$

[ ]: