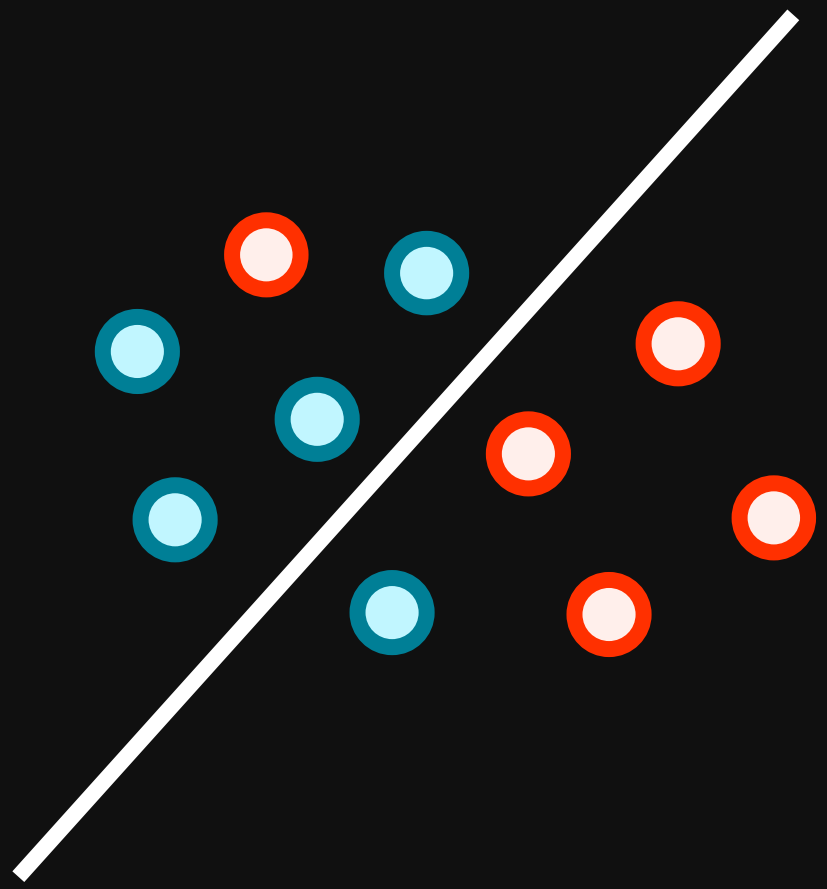
Basics of Machine Learning

Dmitry Ryabokon, github.com/dryabokon



Lesson 03
Data Manipulation

Data Manipulation

Summary

- Create
- Access
- Insert
- Delete
- Inspect
- Aggregate
- Copy
- Order
- Slice
- Subset
- Combine
- Reshape
- IO
- Object creation
- Viewing data
- Selection
- Missing data
- Operations
- Merge
- Grouping
- Reshaping
- Time series
- Categorical
- Plotting
- Getting data in/out

Data Manipulation

Tutorials

- ex_03_numpy.py
- ex_03_pandas.py

Use of Core Python Libs

Create

```
A = numpy.zeros(4)

B = numpy.empty((3, 2))

C = numpy.ones((2, 2))

D = numpy.zeros((2, 3))

E = numpy.full((2, 3), 1)

F = numpy.eye(4)
```

```
G = numpy.full((320, 240), 32, dtype=numpy.uint8)

H = numpy.full((320, 240, 3), 32, dtype=numpy.uint8)

I = numpy.full((320, 240, 3), (0, 0, 200), dtype=numpy.uint8)

J = numpy.array((1, 2, 3))

K = numpy.array((('00', '01', '02'), ('10', '11', '12')))

L = numpy.linspace(10, 25, 9)

M = numpy.arange(10, 25, 5)
```

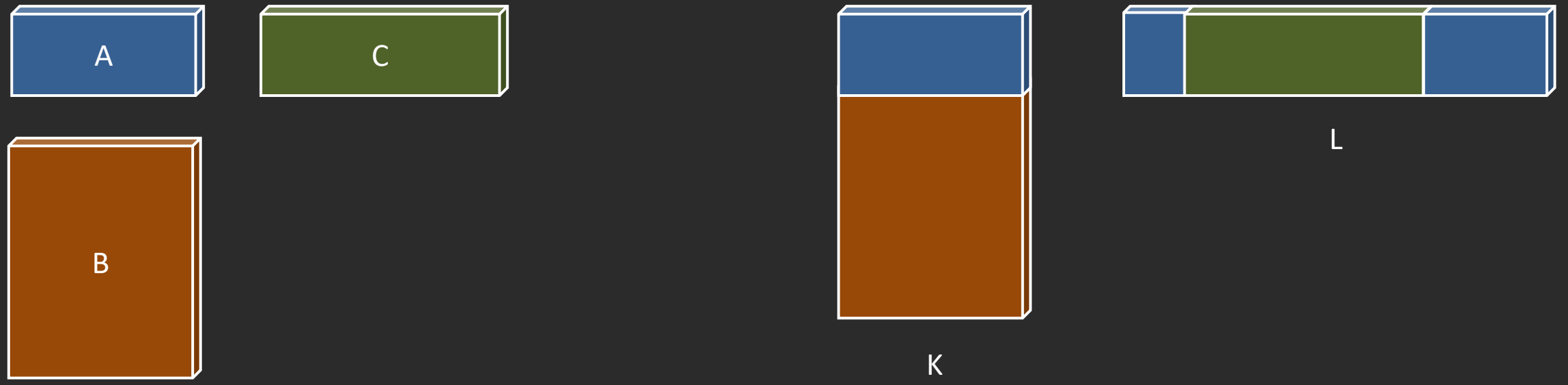
Use of Core Python Libs

Insert

```
A = numpy.full((2, 3), 1)
B = numpy.full((5, 3), 3)
C = numpy.full((2, 4), 2)
```

```
K = numpy.insert(A, [2], B, axis=0)
```

```
L = numpy.insert(A, [1], C, axis=1)
```



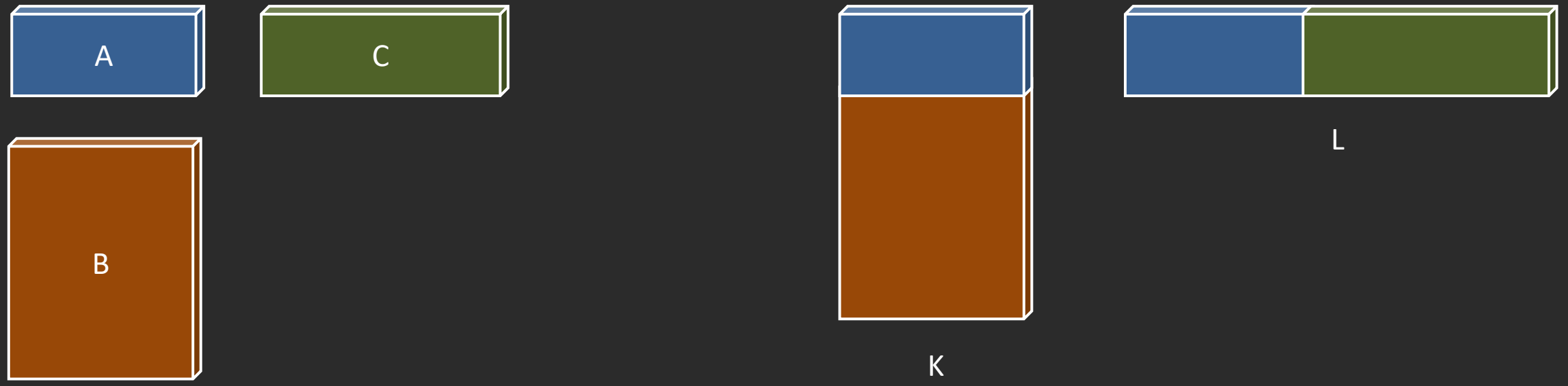
Use of Core Python Libs

Combining

```
A = numpy.full((2, 3), 1)
B = numpy.full((5, 3), 3)
C = numpy.full((2, 4), 2)
```

```
K1 = numpy.vstack((A, B))
K2 = numpy.append(A, B, axis=0)
K3 = numpy.concatenate((A, B), axis=0)
```

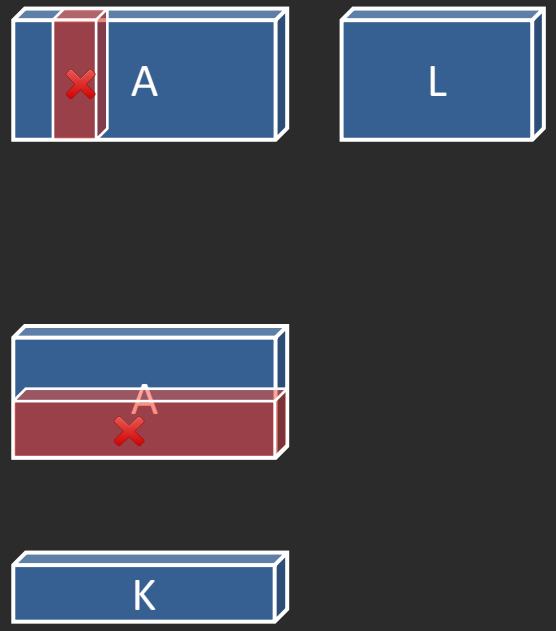
```
L1 = numpy.hstack((A, C))
L2 = numpy.append(A, C, axis=1)
L3 = numpy.concatenate((A, C), axis=1)
```



Use of Core Python Libs

Delete

```
A = numpy.full((2, 3), 1)


K = numpy.delete(A, [1],axis=0)
L = numpy.delete(A, [1],axis=1)
```



Use of Core Python Libs

Inspect

```
def ex_02_inspect():
    A = numpy.full((2, 3), 1)

    sh = A.shape
    ndims = A.ndim
    size = A.size
    the_type = A.dtype
    type_name = A.dtype.name

    return
```

Use of Core Python Libs

Aggregate function

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 0, 0000),
     ('Milk  ', 0, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

AA = A[:, [1, 2]].astype(numpy.int)

A_sum = numpy.sum(AA,axis=0)
A_avg = numpy.mean(AA, axis=0)
A_min = numpy.min(AA, axis=0)
A_max = numpy.max(AA, axis=0)
A_nzr = numpy.count_nonzero(AA, axis=0)
```

Use of Core Python Libs

Copies and Instances

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 0, numpy.nan),
     ('Milk  ', 0, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

B = A.copy()
C = A

B[0,0] = 'Orange'       # Updates B
C[0,0] = 'Peach'        # Updates both A and C (!!)
```

Use of Core Python Libs

Ordering

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 3, 1000),
     ('Milk  ', 7, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

B_fail = numpy.sort(A, axis=0)
C_fail = numpy.sort(A, axis=1)

idx0 = numpy.argsort(A[:, 0])
idx1 = numpy.argsort(A[:, 1])
idx2 = numpy.argsort(A[:, 2])

B = A[idx0]
C = A[idx1]
D = A[idx2]

B2 = numpy.array(sorted(A, key=lambda A: A[0]))
C2 = numpy.array(sorted(A, key=lambda A: A[1]))
D2 = numpy.array(sorted(A, key=lambda A: A[2]))
```

```
[['Apple ' '2' '4000']
 ['Banana' '9' '3000']
 ['Coffee' '7' '6000']
 ['Lemon ' '3' '1000']
 ['Milk  ' '7' '2000']]

[['Apple ' '2' '4000']
 ['Lemon ' '3' '1000']
 ['Milk  ' '7' '2000']
 ['Coffee' '7' '6000']
 ['Banana' '9' '3000']]

[['Lemon ' '3' '1000']
 ['Milk  ' '7' '2000']
 ['Banana' '9' '3000']
 ['Apple ' '2' '4000']
 ['Coffee' '7' '6000']]
```

Use of Core Python Libs

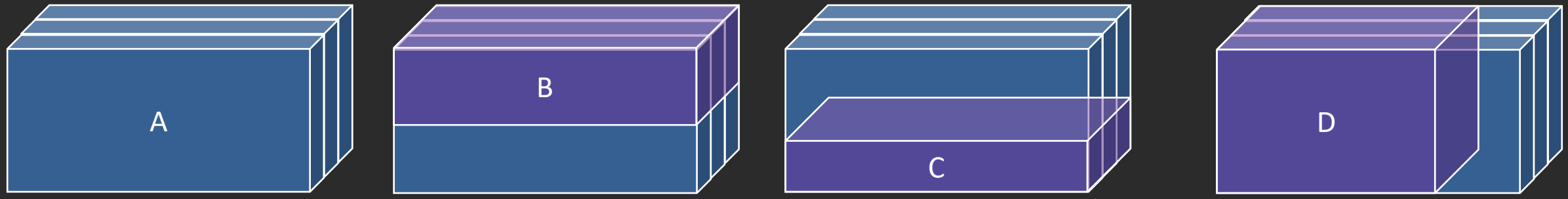
Slicing

```
A = numpy.full((10, 16, 3), 1)
```

```
# (4 x 16 x 3)

B1 = A[0:4]
B2 = A[:4]
```

```
# (2 x 16 x 3)

C1 = A[8:]
C2 = A[10 - 2:]
C3 = A[-2:]
```

```
# (10 x 13 x 3)

D1 = A[:, :13]
D2 = A[:, -3]
```

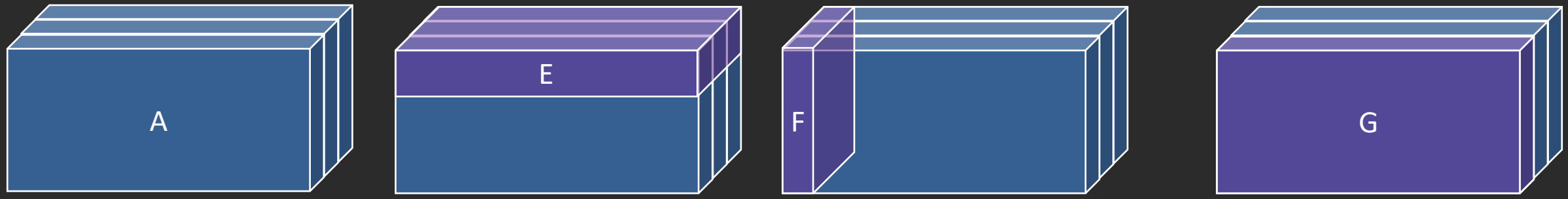


Use of Core Python Libs

Slicing

```
A = numpy.full((10, 16, 3), 1)
```

```
E = A[0, :, :]  # (16,3)
F = A[:, 0, :]  # (10,3)
G = A[:, :, 0]  # (10,16)
```

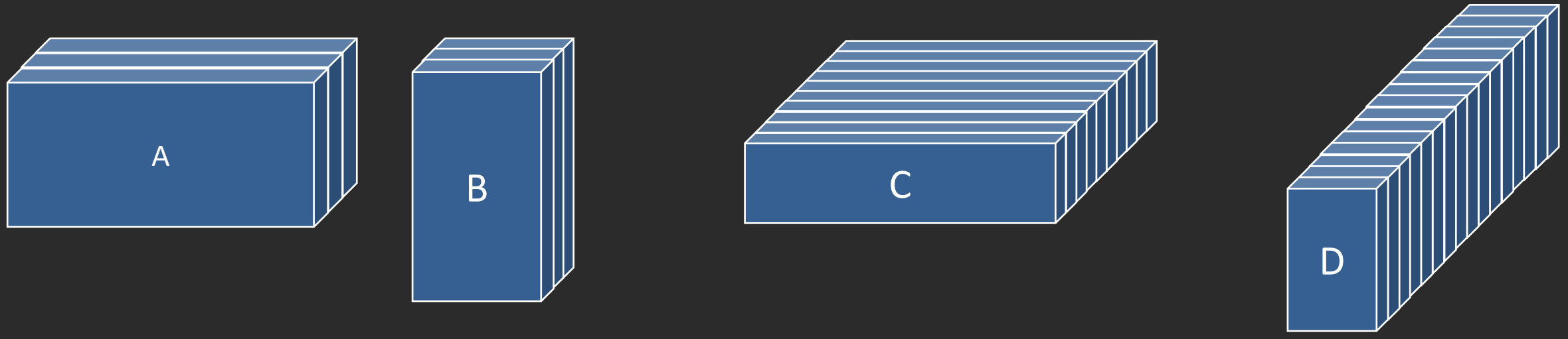


Use of Core Python Libs

Reshape

```
A = numpy.full((10,16,3), 1)
```

```
B = numpy.swapaxes(A,0,1)           # (16 x 10 x 3)
C = numpy.swapaxes(A,0,2)           # ( 3 x 16 x 10)
D = numpy.swapaxes(A,1,2)           # (10 x  3 x 16)

B2 = numpy.transpose(A, (1, 0, 2))  # (16 x 10 x 3)
E  = numpy.transpose(A, (2, 0, 1))  # (3  x 10 x 16)
```



Use of Core Python Libs

IO: binary data

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 0, 0000),
     ('Milk  ', 0, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

AA = A[:, [1, 2]].astype(numpy.int)

numpy.save('./A', A)
B = numpy.load('./A.npy')
```

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 0, 0000),
     ('Milk  ', 0, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

AA = A[:, [1, 2]].astype(numpy.int)

with open('./A.dat', "wb") as f:
    pickle.dump(A, f)

with open('./A.dat', "rb") as f:
    B = pickle.load(f)
```

Use of Core Python Libs

IO: text data

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 0, numpy.nan),
     ('Milk  ', 0, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

data_type = A.dtype

numpy.savetxt('A.txt', A, fmt='%s', delimiter='\t')
B = numpy.loadtxt('A.txt', dtype=data_type, delimiter='\t')
```

```
Apple    2        4000
Lemon    nan      0
Milk     0        2000
Banana   9        3000
Coffee   7        6000
```

Use of Core Python Libs

Ravel

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 3, 1000),
     ('Milk  ', 7, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

F1 = numpy.ravel(A)
F2 = A.flatten()

idx_cr = numpy.unravel_index([2, 4, 5], A.shape)

A[idx_cr] = numpy.nan
```

```
[['Apple ' '2' 'nan']
 ['Lemon ' 'nan' 'nan']
 ['Milk  ' '7' '2000']
 ['Banana' '9' '3000']
 ['Coffee' '7' '6000']]
```

Use of Core Python Libs

Print options

```
A = numpy.array([[1.00002]])
print(A)

numpy.set_printoptions(precision=3)
print(A)
```

```
[[1.00002]]
[[1.]]
```

Use of Core Python Libs

NaN

```
A = numpy.zeros((2, 2))
A[0, 0] = numpy.nan

mask_is_nan = numpy.isnan(A)
A_has_any_nan = numpy.any(numpy.isnan(A))
```

```
In [11]: None == None # noqa: E711
Out[11]: True

In [12]: np.nan == np.nan
Out[12]: False
```

Pandas

Use of Core Python Libs

Create

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 3, 1000),
     ('Lemon ', 9, 7000),
     ('Milk  ', 7, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

df = pd.DataFrame(data=A, index=None, columns=['Product', '#', 'Price'])
df = df.astype({'#': 'int32','Price': 'int32'})
```

Use of Core Python Libs

Inspect

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 3, 1000),
     ('Lemon ', 9, 7000),
     ('Milk  ', 7, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

df = pd.DataFrame(data=A, index=None, columns=['Product', '#', 'Price'])
df = df.astype({'#': 'int32','Price': 'int32'})


print('--------HEAD--------')
print(df.head())
print('\n--------TAIL--------')
print(df.tail())
```

```
--------HEAD--------
  Product  #  Price
0  Apple   2   4000
1  Lemon   3   1000
2  Lemon   9   7000
3  Milk    7   2000
4  Banana  9   3000

--------TAIL--------
  Product  #  Price
1  Lemon   3   1000
2  Lemon   9   7000
3  Milk    7   2000
4  Banana  9   3000
5  Coffee  7   6000
```

Use of Core Python Libs

Inspect: columns

```
['Product' '#' 'Price']
```

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 3, 1000),
     ('Lemon ', 9, 7000),
     ('Milk  ', 7, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

df = pd.DataFrame(data=A, index=None, columns=['Product', '#', 'Price'])
df = df.astype({'#': 'int32','Price': 'int32'})



columns = df.columns.to_numpy()
print(columns)
```

Use of Core Python Libs

Inspect: index

```
rows, cols = 10, 3
idx_dates = pd.date_range("20210101", periods=rows)
columns = [chr(ord('A') + c) for c in range(cols)]
A = (99 * numpy.random.random((rows, cols))).astype(int)
df = pd.DataFrame(data=A, index=idx_dates, columns=columns)
```

```
21-01-01
21-01-02
21-01-03
21-01-04
21-01-05
21-01-06
21-01-07
21-01-08
21-01-09
21-01-10
```

```
idx = df.index.to_numpy() # str

idx2 = (pd.to_datetime(idx).strftime('%y-%m-%d')).to_numpy()    # datetime
```

Use of Core Python Libs

Slicing: columns

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 3, 1000),
     ('Lemon ', 9, 7000),
     ('Milk  ', 7, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

df = pd.DataFrame(data=A, index=None, columns=['Product', '#', 'Price'])
df = df.astype({'#': 'int32','Price': 'int32'})


df_sliced1 = df[['Product', '#']]
print(df_sliced1)
print()

df_sliced2 = df.loc[:,['Product', '#']]
print(df_sliced2)
print()

df_sliced3 = df.iloc[:, [0, 1]]
print(df_sliced3)
```

```
  Product  #
0  Apple   2
1  Lemon   3
2  Lemon   9
3  Milk    7
4  Banana  9
5  Coffee  7

  Product  #
0  Apple   2
1  Lemon   3
2  Lemon   9
3  Milk    7
4  Banana  9
5  Coffee  7

  Product  #
0  Apple   2
1  Lemon   3
2  Lemon   9
3  Milk    7
4  Banana  9
5  Coffee  7
```

Use of Core Python Libs

Slicing: rows

```
A = numpy.array(
    (('Apple ', 2, 4000),
     ('Lemon ', 3, 1000),
     ('Lemon ', 9, 7000),
     ('Milk  ', 7, 2000),
     ('Banana', 9, 3000),
     ('Coffee', 7, 6000)))

df = pd.DataFrame(data=A, index=None, columns=['Product', '#', 'Price'])
df = df.astype({'#': 'int32','Price': 'int32'})


df_sliced1 = df[2:4]
print(df_sliced1)
print()

df_sliced2 = df.loc[2:4]
print(df_sliced2)
print()

df_sliced3 = df.iloc[2:4]
print(df_sliced3)
```

```
  Product  #  Price
2  Lemon   9   7000
3  Milk    7   2000

  Product  #  Price
2  Lemon   9   7000
3  Milk    7   2000
4  Banana  9   3000

  Product  #  Price
2  Lemon   9   7000
3  Milk    7   2000
```

Use of Core Python Libs

Slicing

```
time_range = df.index.to_numpy()
columns = df.columns.to_numpy()
columns_filtered = columns[[0,1]]

print('\n'+'-' * 32 + '\nslice over specific time')
print(df.loc[time_range[:3], :])

print('\n'+'-'*32 + '\nslice over selected columns')
print(df.loc[:, columns_filtered])

print('\n'+'-' * 32 + '\nslice over specific time and columns')
print(df.loc[time_range[1:3], columns_filtered])

print('\n'+'-' * 32 + '\nslice over specific time and columns')
print(df.iloc[1:3, [0,1]])
```

```
slice over specific time
              A   B   C
2021-01-01   29  35  50
2021-01-02   15  17  28
2021-01-03   12  68  54

--------------------------------
slice over selected columns
              A   B
2021-01-01   29  35
2021-01-02   15  17
2021-01-03   12  68
2021-01-04   70  94
2021-01-05   68  50
2021-01-06   31  78
2021-01-07   97  43
2021-01-08   78  86
2021-01-09    4  73
2021-01-10   78  28

--------------------------------
slice over specific time and columns
              A   B
2021-01-02   15  17
2021-01-03   12  68
```

Use of Core Python Libs

Order

```
time_range = df.index.to_numpy()
columns = df.columns.to_numpy()
columns_filtered = columns[[0,1]]

print('\n'+'-' * 32 + '\nslice over specific time')
print(df.loc[time_range[:3], :])

print('\n'+'-'*32 + '\nslice over selected columns')
print(df.loc[:, columns_filtered])

print('\n'+'-' * 32 + '\nslice over specific time and columns')
print(df.loc[time_range[1:3], columns_filtered])

print('\n'+'-' * 32 + '\nslice over specific time and columns')
print(df.iloc[1:3, [0,1]])
```

```
  Product  #  Price
0   Apple   2   4000
4  Banana   9   3000
5  Coffee   7   6000
1   Lemon   3   1000
2   Lemon   9   7000
3    Milk   7   2000

--------------------
   Product  #  Price
2   Lemon   9   7000
4  Banana   9   3000
3    Milk   7   2000
5  Coffee   7   6000
1   Lemon   3   1000
0   Apple   2   4000

--------------------
   Product  #  Price
2   Lemon   9   7000
5  Coffee   7   6000
0   Apple   2   4000
4  Banana   9   3000
3    Milk   7   2000
1   Lemon   3   1000
```

Use of Core Python Libs

Aggregates

```
col_label = df.columns.to_numpy()[idx_agg]
df_agg = df.groupby(col_label).sum()
print(df_agg)
```

Product	#	Price
Apple	2	4000
Banana	9	3000
Coffee	7	6000
Lemon	12	8000
Milk	7	2000

Use of Core Python Libs

Hashing

```
print(df[['sex']].head())
print()
sex = {'male': 0, 'female': 1}
df['sex'] = df['sex'].map(sex)
print(df[['sex']].head())
```

```
  sex
0    male
1  female
2  female
3  female
4    male

   sex
0    0
1    1
2    1
3    1
4    0
```

Use of Core Python Libs

Hashing

```
df_res = df.copy()

col_types = numpy.array([str(t) for t in df.dtypes])
are_categoirical = \
    numpy.array([cc in ['object', 'category', 'bool']
                 for cc in col_types])
C = numpy.arange(0, df.shape[1])[are_categoirical]

columns = df.columns.to_numpy()
for column in columns[C]:
    vv = df.loc[:, column].dropna()

    keys = numpy.unique(vv.to_numpy())
    values = numpy.arange(0,len(keys))
    dct = dict(zip(keys,values))
    df_res[column] = df[column].map(dct)
    df_res = df_res.astype({column: 'int32'})
```

```
 survived  pclass     sex   age  ...  deck  embark_town  alive  alone
1         1       1  female  38.0  ...     C    Cherbourg    yes  False
3         1       1  female  35.0  ...     C  Southampton    yes  False
6         0       1    male  54.0  ...     E  Southampton     no   True
10        1       3  female   4.0  ...     G  Southampton    yes  False
11        1       1  female  58.0  ...     C  Southampton    yes   True

[5 rows x 15 columns]

    survived  pclass  sex   age  ...  deck  embark_town  alive  alone
1         1       1    0  38.0  ...     2            0      1      0
3         1       1    0  35.0  ...     2            2      1      0
6         0       1    1  54.0  ...     4            2      0      1
10        1       3    0   4.0  ...     6            2      1      0
11        1       1    0  58.0  ...     2            2      1      1
```

Use of Core Python Libs

Display precision

```
A = numpy.array([[1.00002]])
df = pd.DataFrame(A)
print(df)
print()

pd.set_option("display.precision", 2)
print(df)
```

```
          0
0  1.00002

     0
0  1.0
```

Use of Core Python Libs

IO: read

```
df = pd.read_csv('A.txt', sep='/t')
A_numpy = df.values
```

Use of Core Python Libs

IO: write

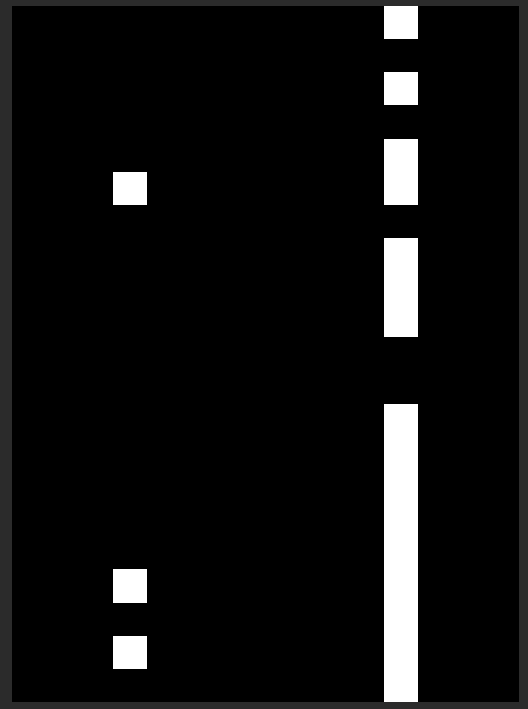
```
df.to_csv(folder_out + 'temp.csv', index=False, sep='\t')
```

Use of Core Python Libs

Is null

```
cv2.imwrite(folder_out + 'nans.png', 255 * (df.isnull()).to_numpy())
```

survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	3	male	22	1	0	7.25	S	Third	man	TRUE		Southampton	no	FALSE
1	1	female	38	1	0	71.2833	C	First	woman	FALSE	C	Cherbourg	yes	FALSE
1	3	female	26	0	0	7.925	S	Third	woman	FALSE		Southampton	yes	TRUE
1	1	female	35	1	0	53.1	S	First	woman	FALSE	C	Southampton	yes	FALSE
0	3	male	35	0	0	8.05	S	Third	man	TRUE		Southampton	no	TRUE
0	3	male		0	0	8.4583	Q	Third	man	TRUE		Queenstown	no	TRUE
0	1	male	54	0	0	51.8625	S	First	man	TRUE	E	Southampton	no	TRUE
0	3	male	2	3	1	21.075	S	Third	child	FALSE		Southampton	no	FALSE
1	3	female	27	0	2	11.1333	S	Third	woman	FALSE		Southampton	yes	FALSE
1	2	female	14	1	0	30.0708	C	Second	child	FALSE		Cherbourg	yes	FALSE
1	3	female	4	1	1	16.7	S	Third	child	FALSE	G	Southampton	yes	FALSE
1	1	female	58	0	0	26.55	S	First	woman	FALSE	C	Southampton	yes	TRUE
0	3	male	20	0	0	8.05	S	Third	man	TRUE		Southampton	no	TRUE
0	3	male	39	1	5	31.275	S	Third	man	TRUE		Southampton	no	FALSE
0	3	female	14	0	0	7.8542	S	Third	child	FALSE		Southampton	no	TRUE
1	2	female	55	0	0	16	S	Second	woman	FALSE		Southampton	yes	TRUE
0	3	male	2	4	1	29.125	Q	Third	child	FALSE		Queenstown	no	FALSE
1	2	male		0	0	13	S	Second	man	TRUE		Southampton	yes	TRUE
0	3	female	31	1	0	18	S	Third	woman	FALSE		Southampton	no	FALSE
1	3	female		0	0	7.225	C	Third	woman	FALSE		Cherbourg	yes	TRUE
0	2	male	35	0	0	26	S	Second	man	TRUE		Southampton	no	TRUE



Use of Core Python Libs

References

- Numpy_Python_Cheat_Sheet.pdf
- https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html