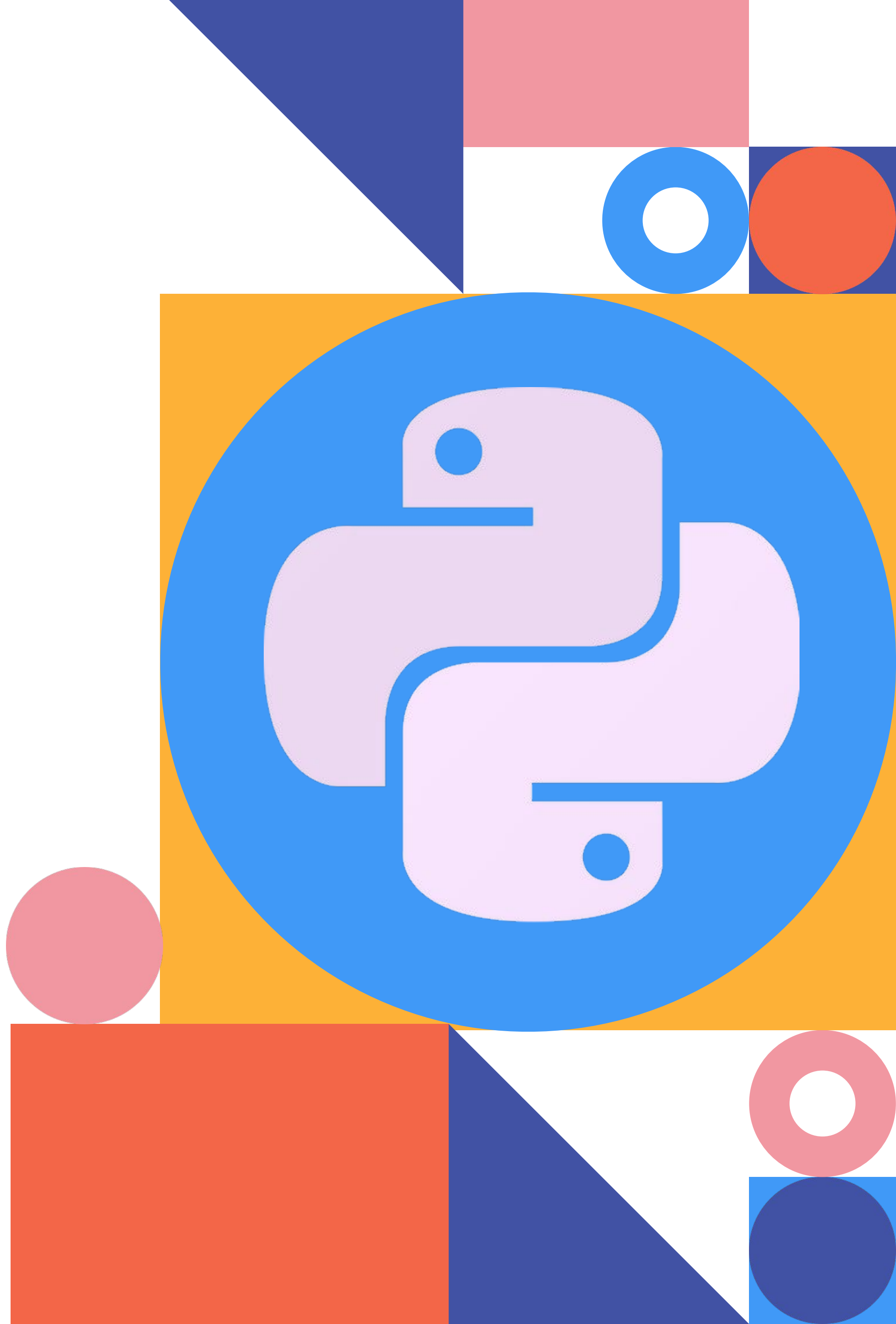
Data Analysis

Yan-Fu Kuo
Dept. of Biomechatronics Engineering
National Taiwan University

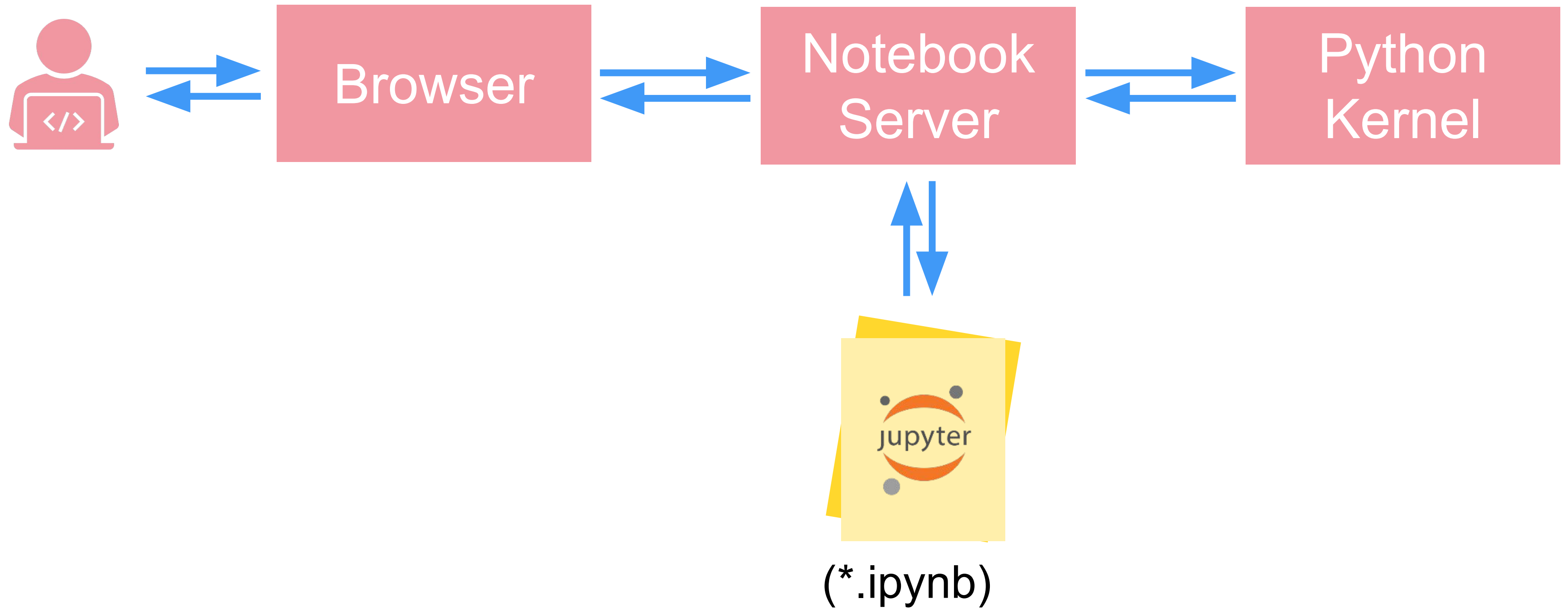
→



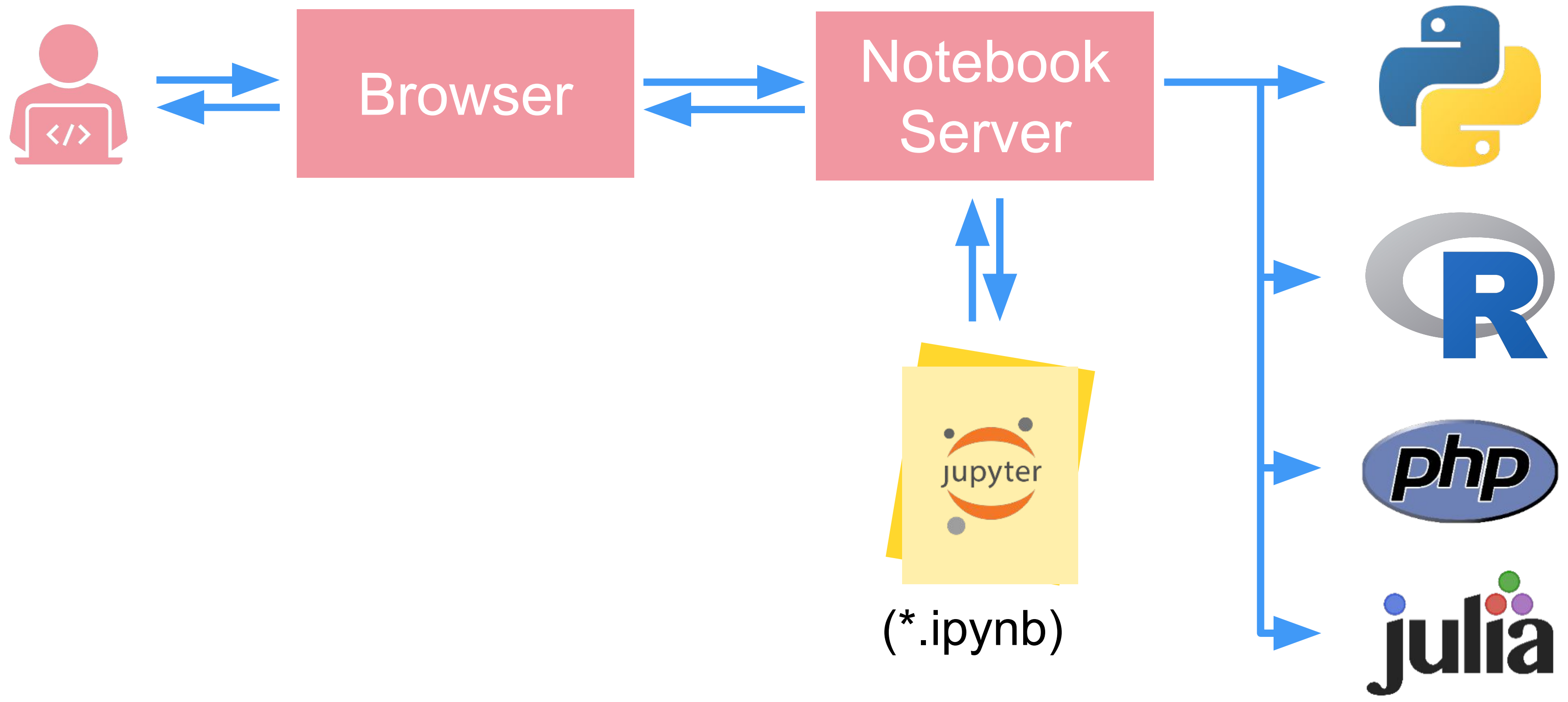
Contents

01 Jupyter Notebook

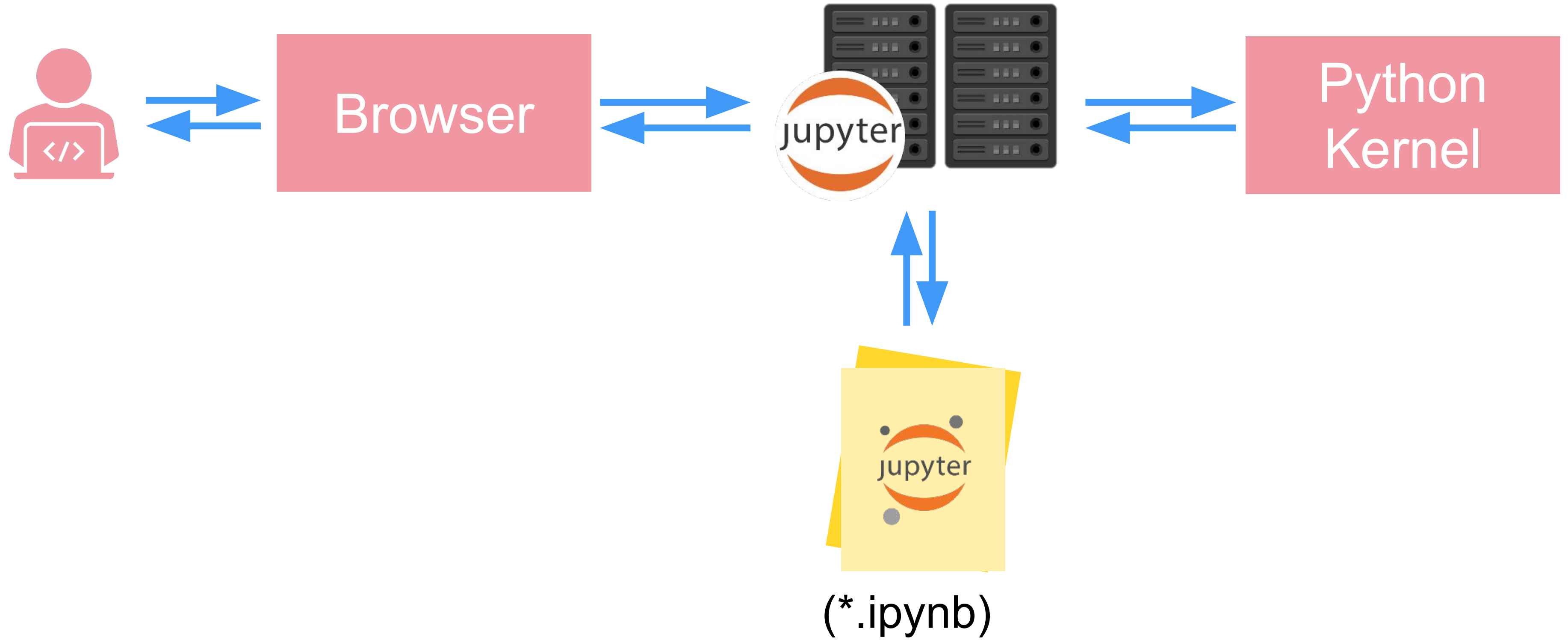
Jupyter



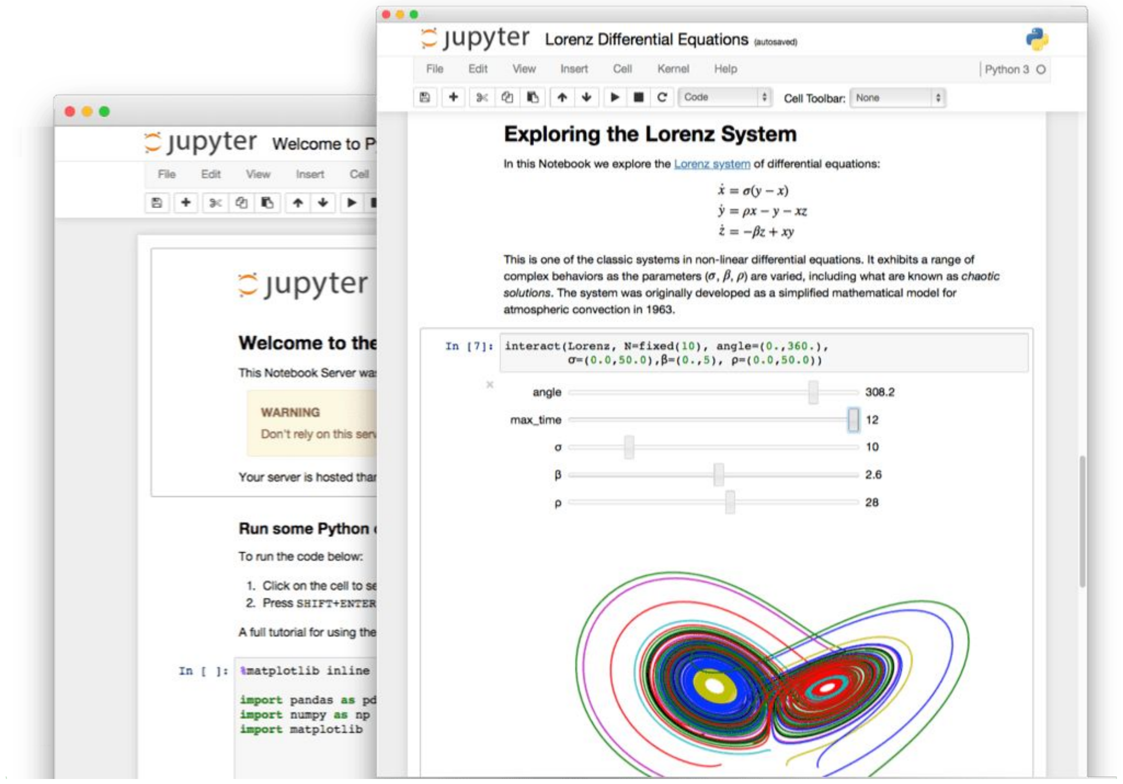
Jupyter Advantages - 1



Jupyter Advantages - 2



Jupyter Advantages - 3


jupyter Welcome to P
File Edit View Insert Cell
jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None

Welcome to the

This Notebook Server wa

WARNING

Don't rely on this ser

Your server is hosted tha

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

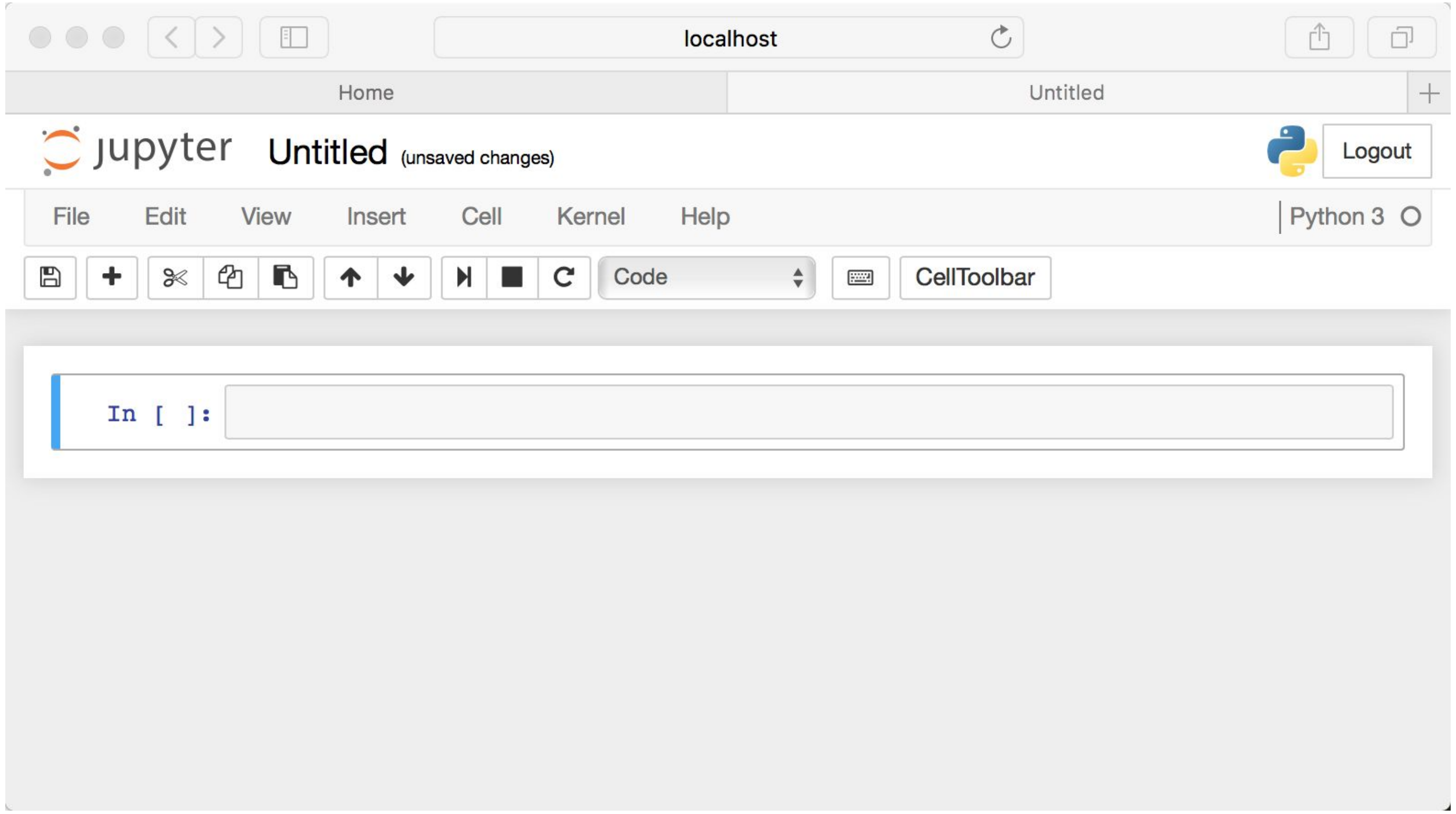
```
angle     308.2
max_time  12
σ         10
β         2.6
ρ         28
```

Text

Code

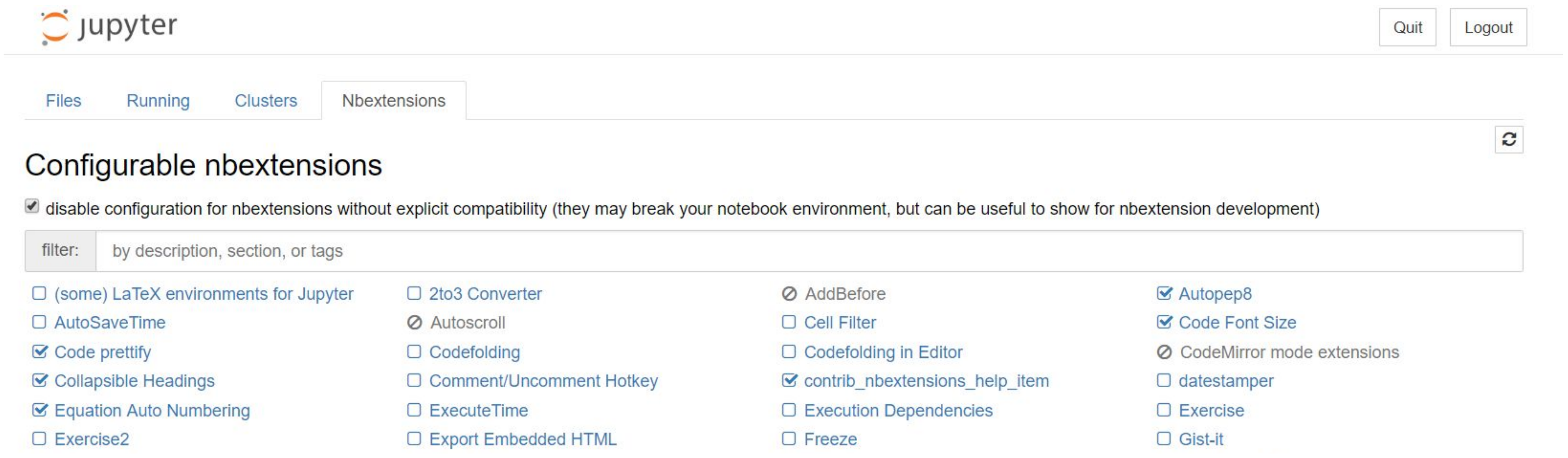
Output

Jupyter Interface



Extensions

```
$ pip3 install jupyter_contrib_nbextensions
$ jupyter contrib nbextension install
```



Configurable nbextensions

☑ disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

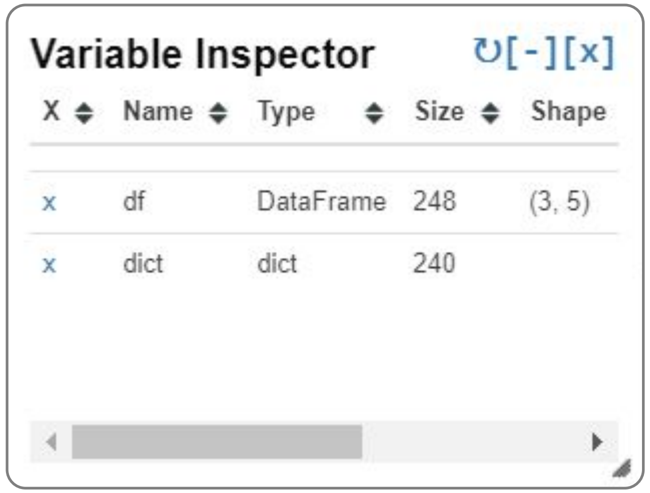
filter: by description, section, or tags

☐ (some) LaTeX environments for Jupyter	☐ 2to3 Converter	⊘ AddBefore	☑ Autopep8
☐ AutoSaveTime	⊘ Autoscroll	☐ Cell Filter	☑ Code Font Size
☑ Code prettify	☐ Codefolding	☐ Codefolding in Editor	⊘ CodeMirror mode extensions
☑ Collapsible Headings	☐ Comment/Uncomment Hotkey	☑ contrib_nbextensions_help_item	☐ datestamper
☑ Equation Auto Numbering	☐ ExecuteTime	☐ Execution Dependencies	☐ Exercise
☐ Exercise2	☐ Export Embedded HTML	☐ Freeze	☐ Gist-it

https://towardsdatascience.com/jupyter-notebook-extensions-517fa69d2231

Which to Use

- Table of Contents: easier navigation
- Autopep8: neat code in one click
- Variable inspector: keep track of your workspace
- ExecuteTime: show when and how long cells ran
- Hide Code input: hide the work show the results


Variable Inspector ↻[-][x]

X	Name	Type	Size	Shape
x	df	DataFrame	248	(3, 5)
x	dict	dict	240	

02 Useful Build-in Modules

os Modules

```python
import os
```

mkdir

cp

mv
(rename)

rm

os Modules

os.getcwd()
os.chdir()
os.listdir()

os.mkdir()
os.makedirs()
os.rmdir()
os.remove()
os.rename()

os.walk()

`os.path` Sub-module

`.isdir()`

`.isfile()`

`.join()`

glob

shutil Modules

```python
import shutil
```

mkdir

cp

mv
(rename)

rm

Copying and Moving Files

```
shutil.copyfile(src, dst)
```

Copy the contents (no metadata) of the file named src to a file named dst. dst must be the complete target file name; look at shutil.copy() for a copy that accepts a target directory path. If src and dst are the same files, Error is raised. The destination location must be writable; otherwise, an IOError exception will be raised. If dst already exists, it will be replaced. Special files such as character or block devices and pipes cannot be copied with this function. src and dst are path names given as strings.

```
shutil.copy(src, dst)
```

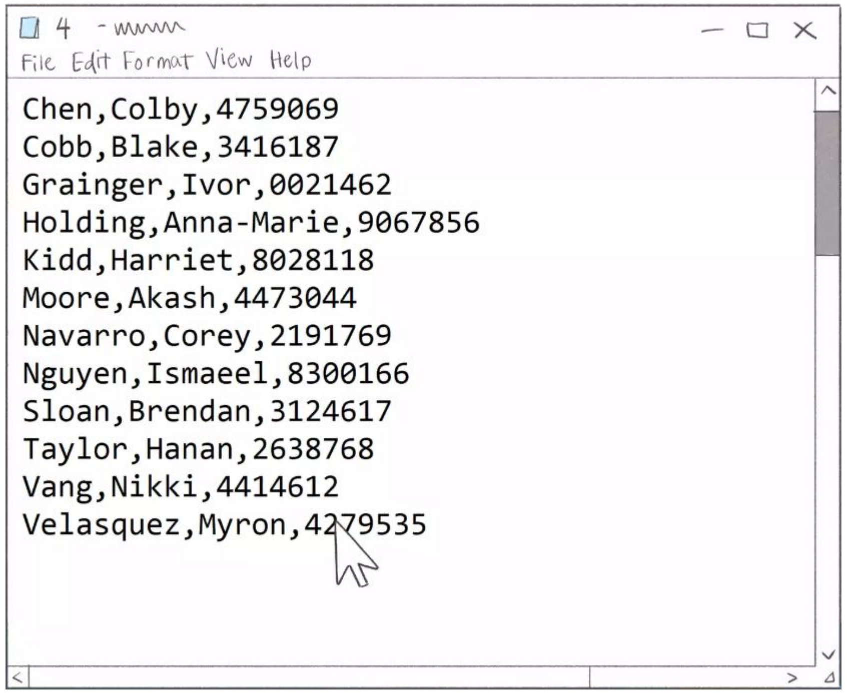
Copy the file src to the file or directory dst. If dst is a directory, a file with the same basename as src is created (or overwritten) in the directory specified. Permission bits are copied. src and dst are path names given as strings.

```
shutil.move(src, dst)
```

Recursively move a file or directory (*src*) to another location (*dst*).

`csv` Modules

A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values.



```
Chen,Colby,4759069
Cobb,Blake,3416187
Grainger,Ivor,0021462
Holding,Anna-Marie,9067856
Kidd,Harriet,8028118
Moore,Akash,4473044
Navarro,Corey,2191769
Nguyen,Ismaeel,8300166
Sloan,Brendan,3124617
Taylor,Hanan,2638768
Vang,Nikki,4414612
Velasquez,Myron,4279535
```

```python
import csv
```

Reading a CSV

`csv.reader()` – Reading data from the csv file

The reader can be used as an iterator to process

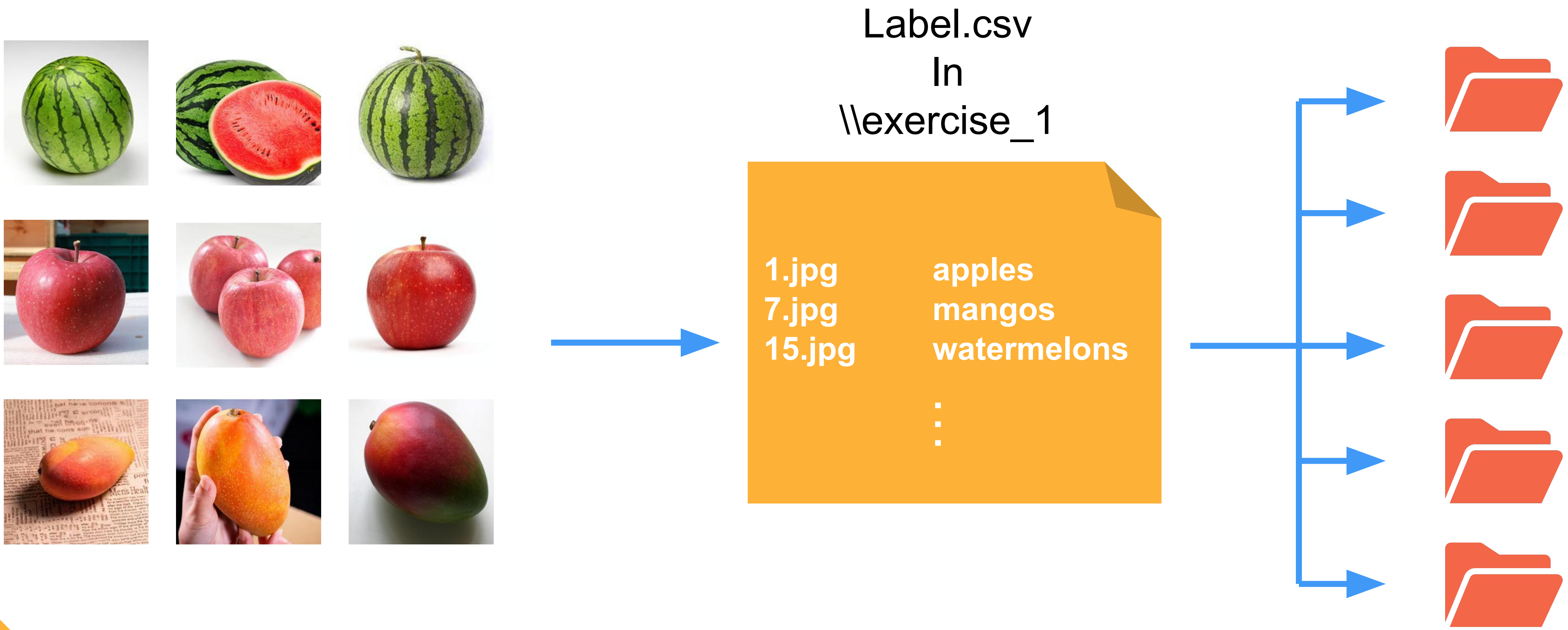
```python
import csv

f = open('titanic.csv')
csv_reader = csv.reader(f, delimiter=',')
for row in csv_reader:
    print(row)
f.close()
```

Writing a CSV

`csv.writer()` – Writing a CSV file

```python
with open('employee_file.csv', mode='w') as employee_file:
    employee_writer = csv.writer(employee_file, delimiter=',')

    employee_writer.writerow(['John Smith', 'Accounting', 'November'])
    employee_writer.writerow(['Erica Meyers', 'IT', 'March'])
```

Exercise 1



Hint: `next()`, `os.makedirs()`, `shutil.copy()`

03 Data Analysis Packages: Numpy

NumPy (Numerical Python)

matplotlib

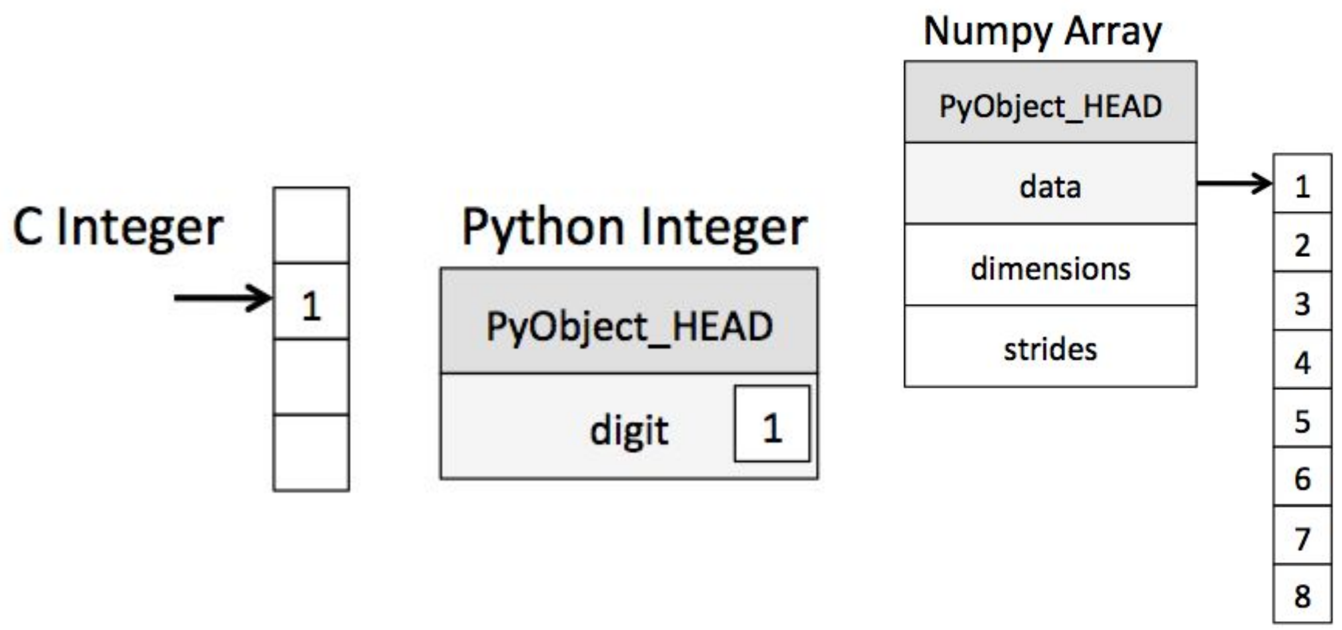
scikits learn
machine learning in Python

pandas

scikit-image
image processing in python

NumPy

Python list vs. NumPy array



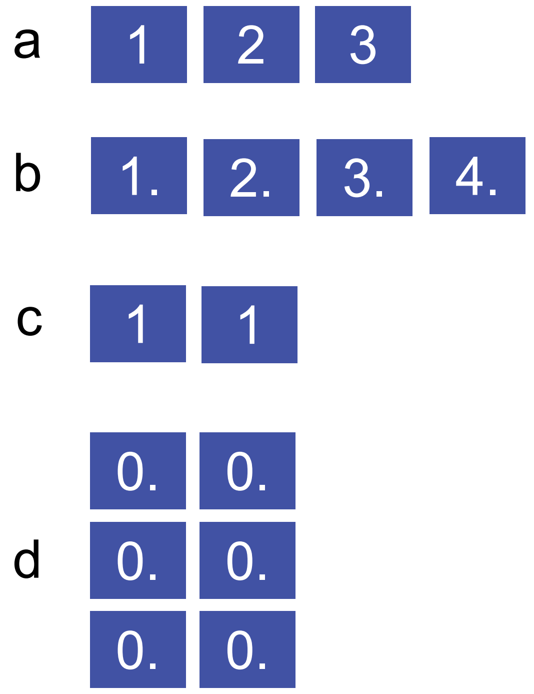
Creating NumPy Array

Creating Arrays from Python Lists

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([1, 2, 3, 4], dtype='float32')
```

Creating Arrays from Scratch

```
c = np.ones(2, dtype=int)
d = np.zeros((3, 2), dtype=float)
```



NumPy Standard Data Types

```
np.zeros(10, dtype='int16')
```

or

```
np.zeros(10, dtype=np.int16)
```

Data type	Description
bool_	Boolean (True or False) stored as a byte
int_	Default integer type (same as C long; normally either int64 or int32)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa

Creating NumPy Array

Try these functions:

```
np.full()       np.random.random()
np.arange()     np.random.normal()
np.linspace()   np.eye()
```

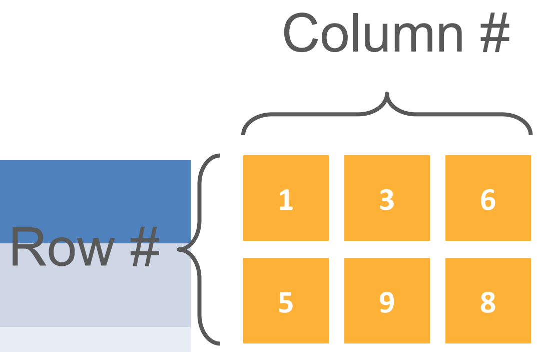
`np.empty()` → What are the values in the array?

What is the reason of using np.empty()?

NumPy Array Class

np

.ndim	the number of axes (dimensions) of the array.
.shape	the dimensions of the array. ←tuple
.size	the total number of elements of the array.
.dtype	an object describing the type of the elements in the array.
.itemsize	the size in bytes of each element of the array.
.data	the buffer containing the actual elements of the array.



ndim = 2

shape = (2, 3)

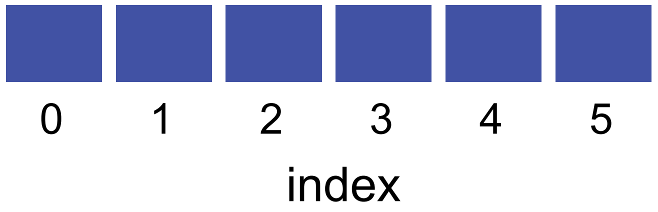
size = 6

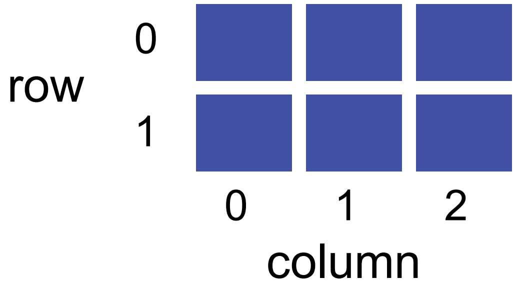
dtype = int32

Array Indexing

```
array[index]
```

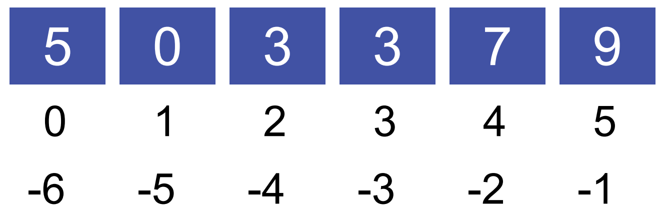
```
array[row, column]
```



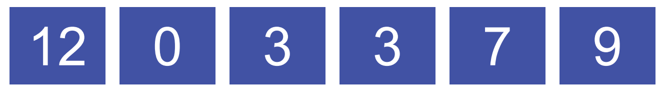


Array Indexing

```
x = np.array([5, 0, 3, 3, 7, 9])

print(x[0]) #5
print(x[-1]) #9

x[0] = 12

x[1] = 3.14
print(x) #???
```



Values can also be modified using any of the index notation



What is the value of x after executing the last command

Fancy Indexing

Accessing multiple array elements at once.

```
rand = np.random.RandomState(42)
x = rand.randint(100, size=10)
print(x)
type(x)

y = x[3], x[7], x[2]
print(y)
type(y)

z = x[np.array([3, 7, 4])]
print(z)
type(z)
```

```
x: [51 92 14 71 60 20 82 86 74 74]
```

```
y: 71, 86, 14
```

```
z: array([71, 86, 60])
```

https://numpy.org/doc/stable/user/basics.indexing.html#advanced-indexing

Array Slicing: 1D

```
array[start:stop:step]
```

default values: `start=0, stop=size of dimension, step=1`

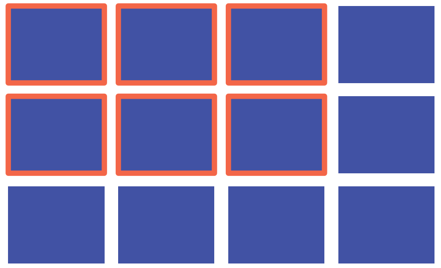
```
x = np.arange(10)
x[:5]  # first five elements
x[5:]  # elements after index 5
x[4:7]  # middle sub-array
x[::2]  # every other element
x[::-1]  # all elements, reversed
```

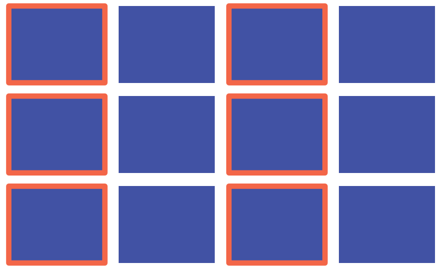
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]
[4, 5, 6]
[0, 2, 4, 6, 8]
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

Array Slicing: 2D

```
array[:2, :3]
```

```
array[:3, ::2]
```



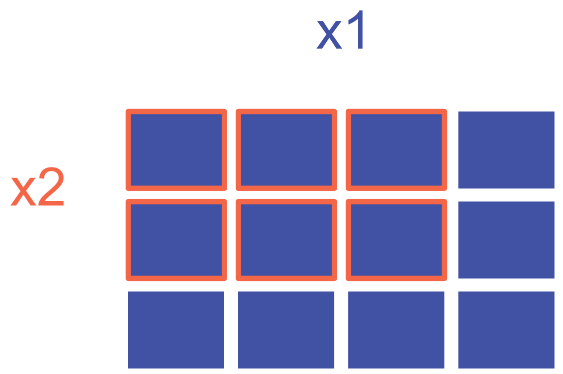


How to access a single row and column?

What is the value of `array[::-1, ::-1]`?

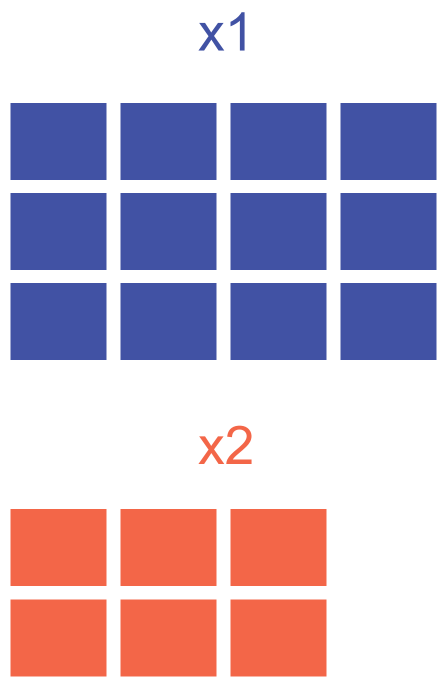
Array Aliasing

```
x1 = np.random.randint(10, size=(3, 4))

x2 = x1[:2, :3]
print(x2)

x2[0, 0] = 99
print(x2)

print(x1)
```



Creating Copies of Arrays

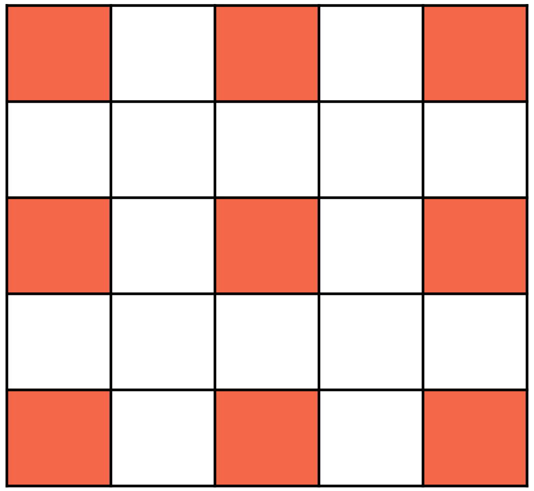
```
x1 = np.random.randint(10, size=(3, 4))

x2 = x1[:2, :2].copy()
print(x2)

x2[0, 0] = 99
print(x2)

print(x1)
```



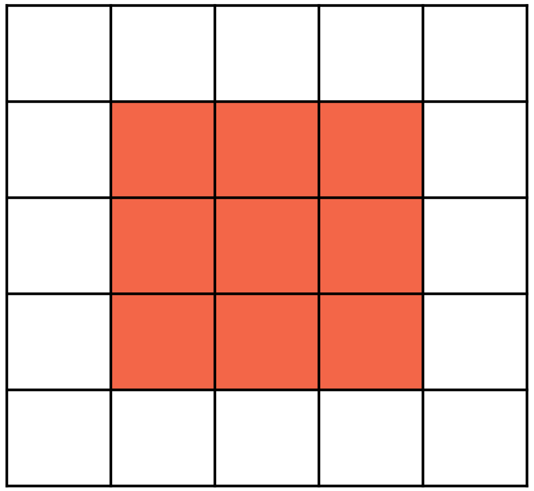
Exercise 2

An M-by-M matrix X is given. Without using loops, extract values from matrix X to create the following:

A composed of all values in odd columns AND odd rows of X

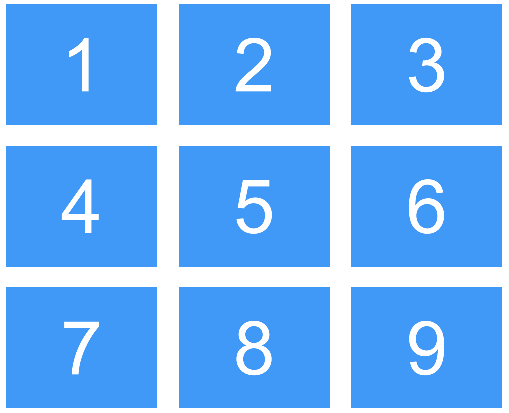


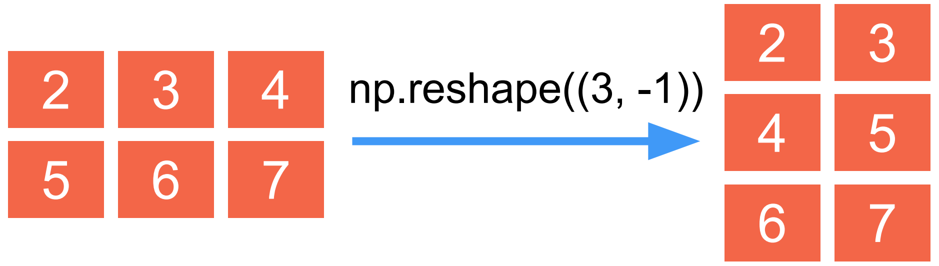
B composed of all entries of X, except for the outside rows and columns

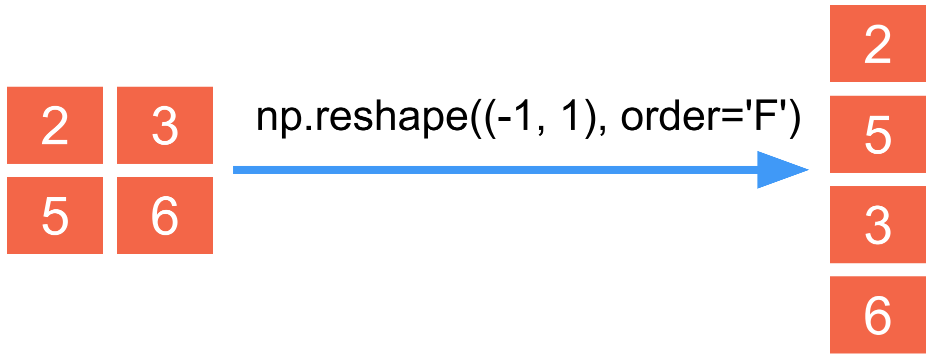


Reshaping of Arrays

```
grid = np.arange(1, 10).reshape((3, 3))
print(grid)
```







https://numpy.org/doc/stable/reference/generated/numpy.reshape.html

Concatenation of Arrays

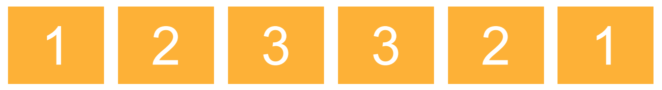
One-dimensional array

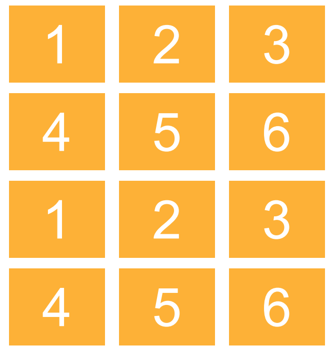
```
x = np.array([1, 2, 3])
y = np.array([3, 2, 1])
np.concatenate([x, y])
```

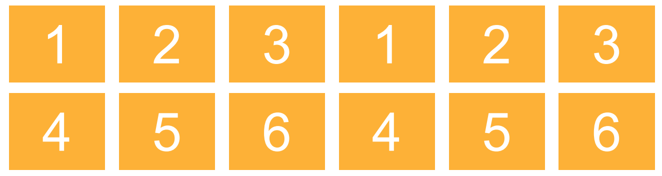
Two-dimensional array

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

```
np.concatenate([grid, grid], axis=1)
```







Array Arithmetic

```python
x = np.arange(4)
print("x     =", x)
print("x + 5 =", x + 5)
print("x - 5 =", x - 5)
print("x * 2 =", x * 2)
print("x / 2 =", x / 2)
print("x // 2 =", x // 2)
```

```
x     = [0 1 2 3]
x + 5 = [5 6 7 8]
x - 5 = [-5 -4 -3 -2]
x * 2 = [0 2 4 6]
x / 2 = [ 0.   0.5  1.   1.5]
x // 2 = [0 0 1 1]
```

Array Arithmetic

```
x = np.arange(4)
-(0.5*x + 1) ** 2
```

```
array([-1.  , -2.25, -4.  , -6.25])
```

```
np.add(x, 2)
```

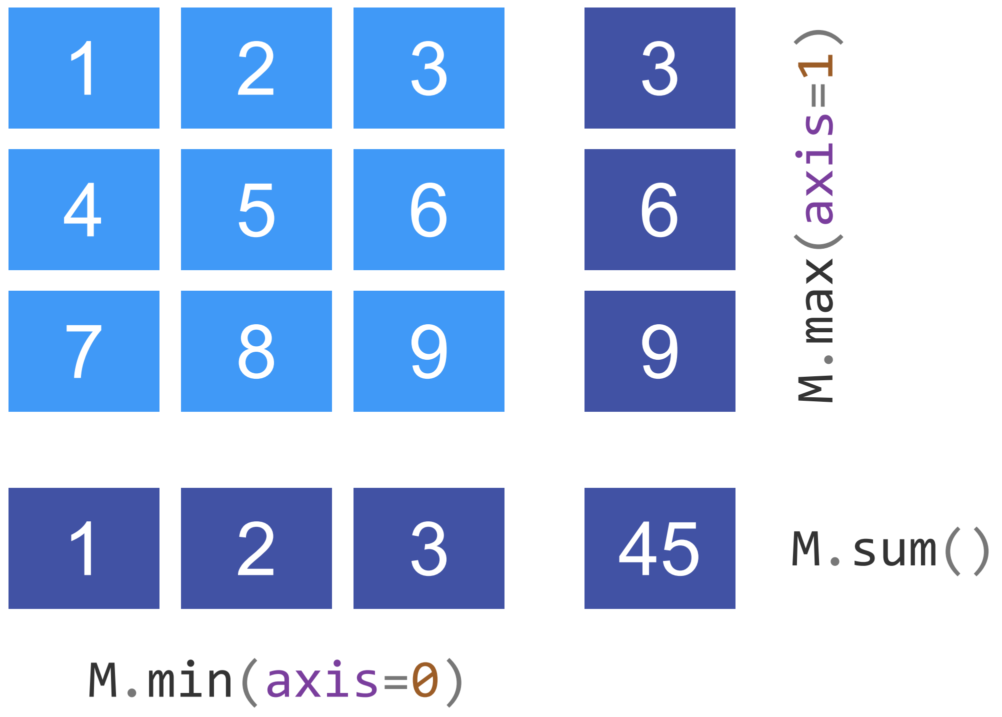
```
array([2, 3, 4, 5])
```

Name	Operator	Equivalent func.
Addition	+	np.add
Subtraction	-	np.subtrat
Multiplication	*	np.multiply
Division	/	np.divide
Modulus	%	np.mod
Exponentiation	**	np.power
Floor Division	//	np.floor_divide

Aggregation Functions

Function Name	NaN-safe Version	Description
np.sum	np.nansum	Compute sum of elements
np.prod	np.nanprod	Compute product of elements
np.mean	np.nanmean	Compute mean of elements
np.std	np.nanstd	Compute standard deviation
np.var	np.nanvar	Compute variance
np.min	np.nanmin	Find minimum value
np.max	np.nanmax	Find maximum value
np.argmin	np.nanargmin	Find index of minimum value
np.argmax	np.nanargmax	Find index of maximum value
np.median	np.nanmedian	Compute median of elements
np.percentile	np.nanpercentile	Compute rank-based statistics of elements
np.any	N/A	Evaluate whether any elements are true
np.all	N/A	Evaluate whether all elements are true

Multi Dimensional Aggregates



Broadcasting

Broadcasting allows binary operations to be performed on arrays of different sizes

```python
a = np.array([0, 1, 2])
b = np.array([5, 5, 5])
a + b
```

```
array([5, 6, 7])
```

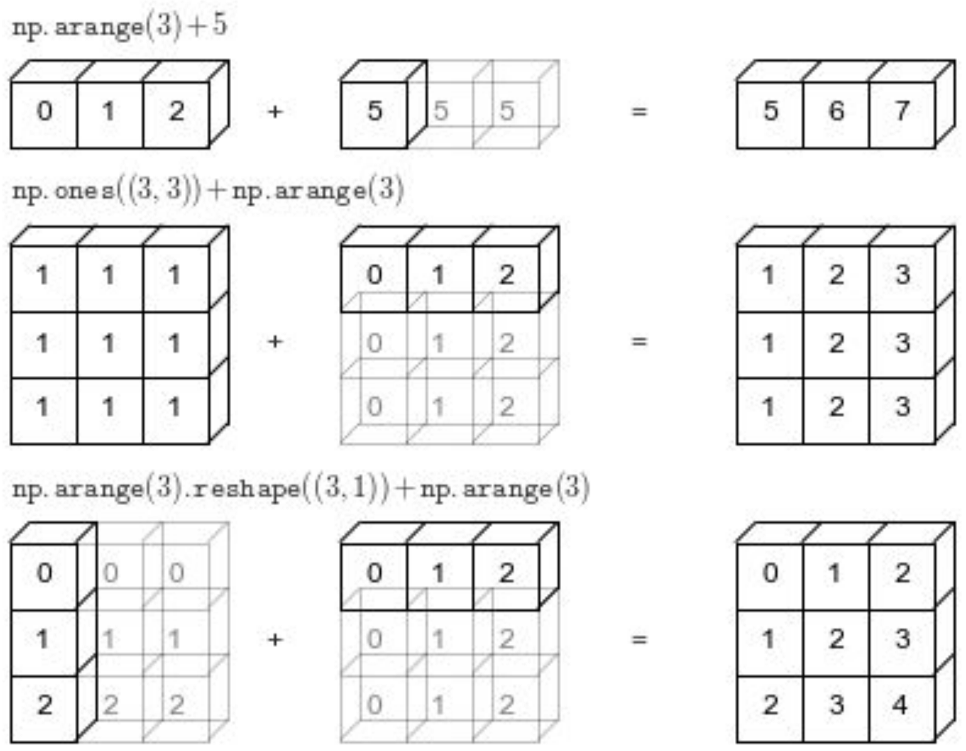
```python
a + 5
```

```
array([5, 6, 7])
```

```python
M = np.ones((3, 3))
M + a
```

```
array([[ 1.,  2.,  3.],
       [ 1.,  2.,  3.],
       [ 1.,  2.,  3.]])
```

Broadcasting



Rules of Broadcasting

Rule 1: If the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded with ones on its leading (left) side.

Rule 2: If the shape of the two arrays does not match in any dimension, the array with shape equal to 1 in that dimension is stretched to match the other shape.

Rule 3: If in any dimension the sizes disagree and neither is equal to 1, an error is raised.

thank you!