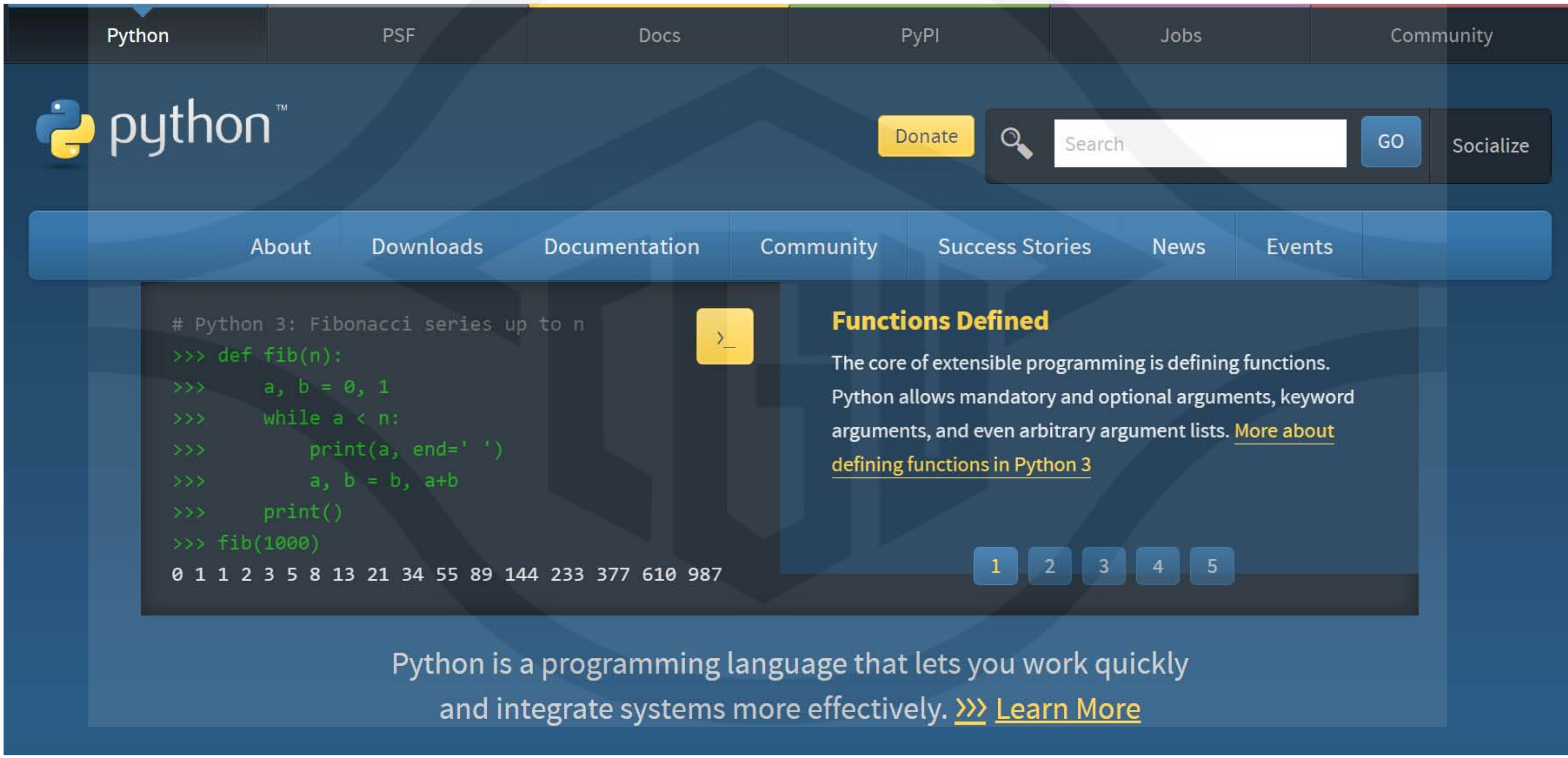
PATTERN RECOGNITION USING PYTHON

Python Basic

Wen-Yen Hsu
Dept Electrical Engineering
Chang Gung University, Taiwan

2019-Spring

Meet Python



https://www.python.org/

TIOBE Programming Community Index

Feb 2019	Feb 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.876%	+0.89%
2	2		C	12.424%	+0.57%
3	**4**		**Python**	**7.574%**	**+2.41%**
4	3		C++	7.444%	+1.72%
5	6		Visual Basic .NET	7.095%	+3.02%
6	8		JavaScript	2.848%	-0.32%
7	5		C#	2.846%	-1.61%
8	7		PHP	2.271%	-1.15%
9	11		SQL	1.900%	-0.46%
10	20		Objective-C	1.447%	+0.32%
11	15		Assembly language	1.377%	-0.46%
12	19		MATLAB	1.196%	-0.03%
13	17		Perl	1.102%	-0.66%
14	9		Delphi/Object Pascal	1.066%	-1.52%
15	13		R	1.043%	-1.04%
16	10		Ruby	1.037%	-1.50%
17	12		Visual Basic	0.991%	-1.19%
18	18		Go	0.960%	-0.46%
19	49		Groovy	0.936%	+0.75%

https://www.tiobe.com/tiobe-index/

The Features of Python

- Rapid Development
  - Interpreted language
  - Dynamic type
  - Readable syntax
  - Sufficient support

Applications

- Web Development
  - Backend framework
  - Web crawler
- GUI Development
- Scientific
  - Artificial intelligence
  - Machine learning
- Embedded System
  - Raspberry Pi

Business Cases of Python

Build Environment

- For Linux or Mac
  - Built in
- For Windows
  - Need to install
    - Pure-Python
    - Distribution package：Anaconda http://docs.continuum.io/anaconda/install/
- Version Problem
  - Python 3 is incompatible with Python 2

Programming Way

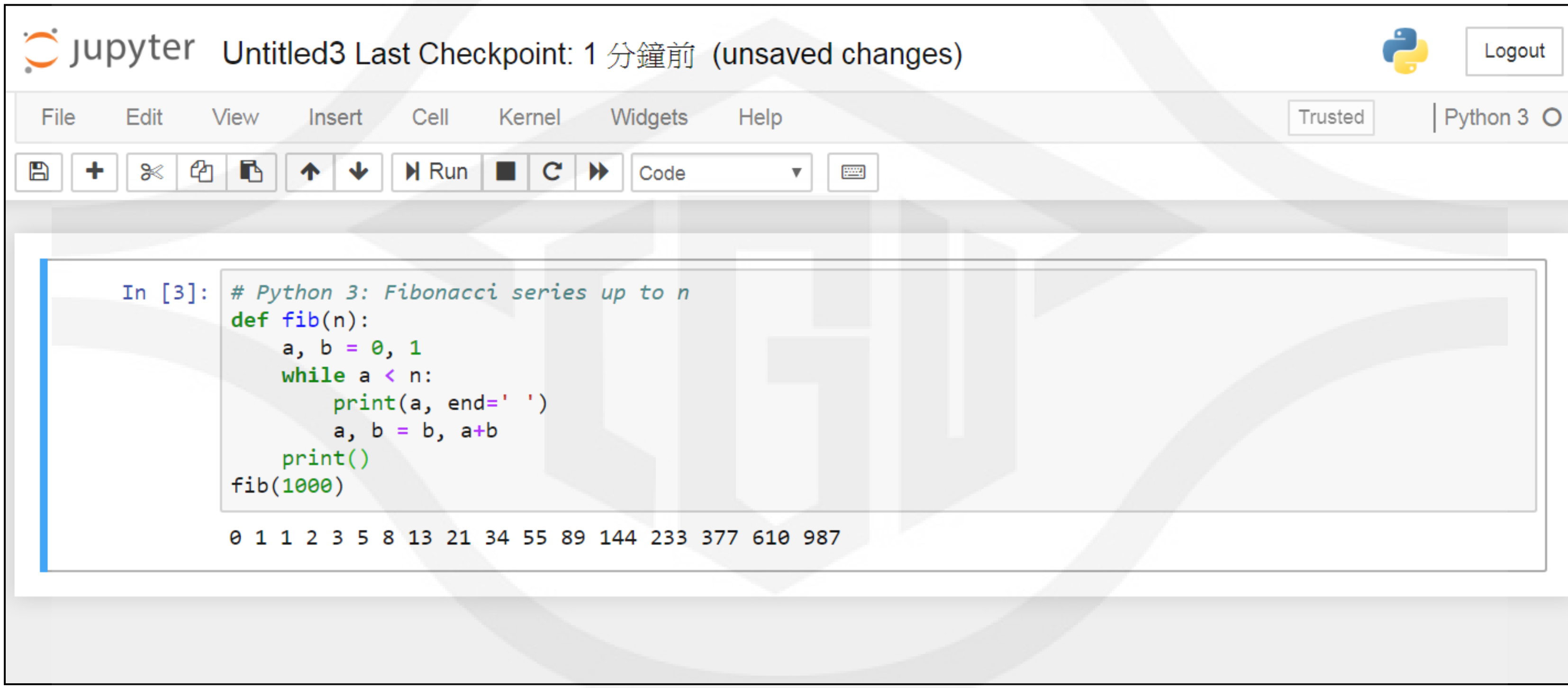
- Command line

系統管理員: 命令提示字元 - python

```
Microsoft Windows [版本 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\CGU>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> # Python 3: Fibonacci series up to n
...
>>> def fib(n):
...     a, b = 0, 1
...     while a < n:
...         print(a, end=' ')
...         a, b = b, a+b
...     print()
...
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>>
```

Programming Way (cont.)

- Run (*.ipynb) in Jupyter Notebook



Shortcut Key in Jupyter Notebook

Key combination	Effect
Shift+Enter	Run this cell and move to next
Ctrl+Enter	Run this cell
Ctrl+/	Comment
Ctrl+]	Increase Indent
Ctrl+[	Decrease Indent
A	Insert cell above
B	Insert cell below
D D	Delete cell

Other Recommended Editors

- PyCharm
- VS code
- Atom
- Notepad++
- Sublime Text

Getting Start

- A simple example about the summation of sequence

```
a = 1
b = 2
c = 3
d = 4

sum = a + b + c + d

print('sum=', sum)
```

Control Flow

- **For** loop in python

```
sum = 0

for i in range(5):
    print('i=', i)
    sum = i + sum

print('sum=', sum)
```

```
sum = 0

for i in range(0, 5, 1):
    print('i=', i)
    sum = i + sum

print('sum=', sum)
```

Encapsulation

- Define a **function** then reuse it

```
def summation(start, end):
    sum = 0
    for i in range(start, end+1, 1):
        sum = i + sum
    return sum

sum_1 = summation(1, 4)
print('sum_1=', sum_1)
sum_2 = summation(2, 7)
print('sum_2=', sum_2)
```

Conditional Statement

- Check **condition** and change behavior

```python
num_1 = 1
num_2 = 3
if num_1 > num_2:
    print('num_1 is greater than num_2')
else:
    print('num_1 is not greater than num_2')
```

Python Modules and Packages

- Numpy (matrix computing)
- SciPy (scientific computing)
- Matplotlib (picture plotting)
- Pandas (data structures)
- Scikit-learn (machine learning)
- PyTorch (deep learning)

general purposes

specific purpose

Import Module

- 3 methods to import module (using numpy as an example)

```python
import numpy
from numpy import array, dot
import numpy as np
```

good choice

- Avoid name conflict issue

```python
import numpy as np

np1 = np.array([1, 2, 3])
np2 = np.array([3, 4, 5])
np3 = np.dot(np1, np2)
print('outcome=', np3)
```

Import Module (cont.)

- Method 2 and Method 3

```python
from numpy import dot, array

np4 = array([1, 2, 3])
np5 = array([3, 4, 5])
#def dot(a, b):
#     c = a*a + b*b
#     return c
np6 = dot(np4, np5)
print('outcome=', np6)
```

```python
import numpy
np7 = numpy.array([1, 2, 3])
np8 = numpy.array([3, 4, 5])
#def dot(a, b):
#     c = a*a + b*b
#     return c
np9 = numpy.dot(np1, np2)
print('outcome=', np9)
```

Danger Zone of Import Module

- **Comment out** the code below and see what happen

```
#def dot(a, b):
#    c = a*a + b*b
#    return c
```

⇩

```
def dot(a, b):
    c = a*a + b*b
    return c
```

Create Vector in Numpy

- We need an array like this:
  vector = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  How to achieve ?

- Direct method

```
import numpy as np
vector_1 = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
print('vector_1=', vector_1)
```

- But, if we need another array like this:
  vector = [0, 1, 2, ...,9486, 9487]

Create Vector (cont.)

- By **np.arange()** method

```
vector_2 = np.arange(10)
print('vector_2=', vector_2)
```

- Using np.arange() method with parameter

```
vector_3 = np.arange(0, 10, 1)
print('vector_3=', vector_3)
```

- Another method similar to np.arange()

```
vector_4 = np.linspace(0, 9, 10)
print('vector_4=', vector_4)
```

What is the difference between these methods ? And the outcome ?

Data Types in Numpy

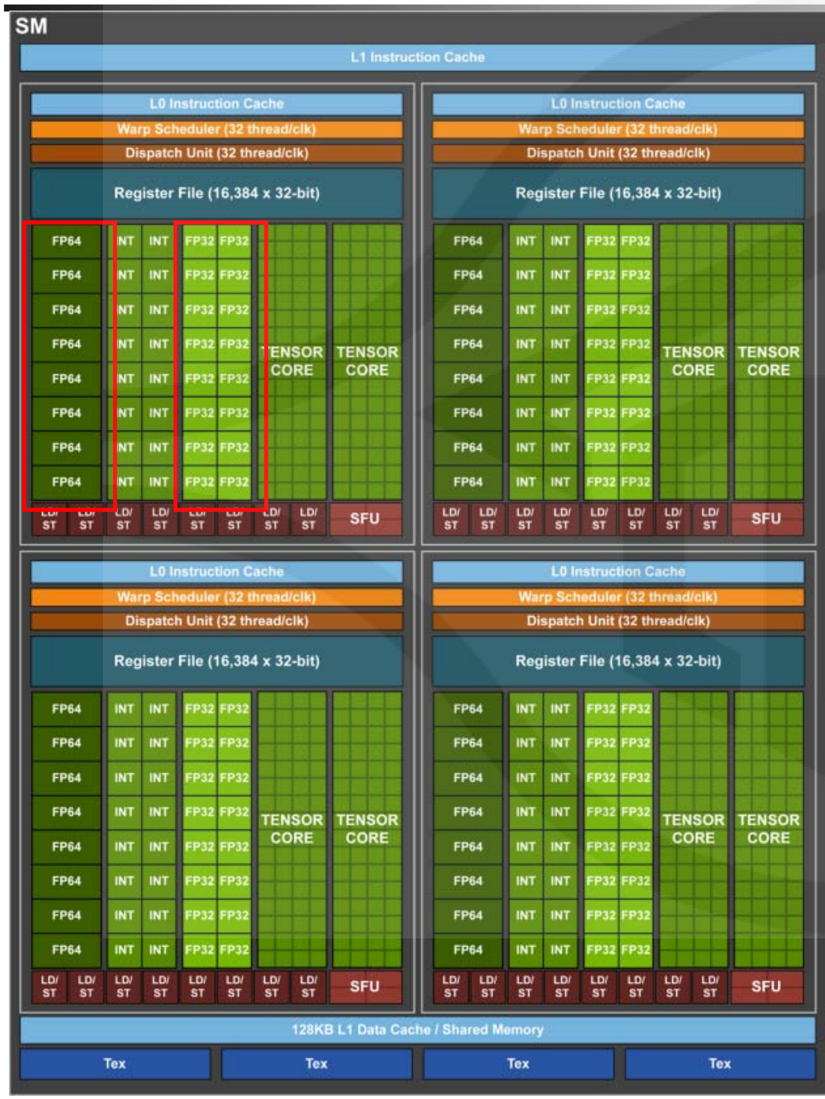
- Define data types

```
vector_3 = np.arange(0, 10, 1, dtype=np.float32)
print('vector_3=', vector_3)
```

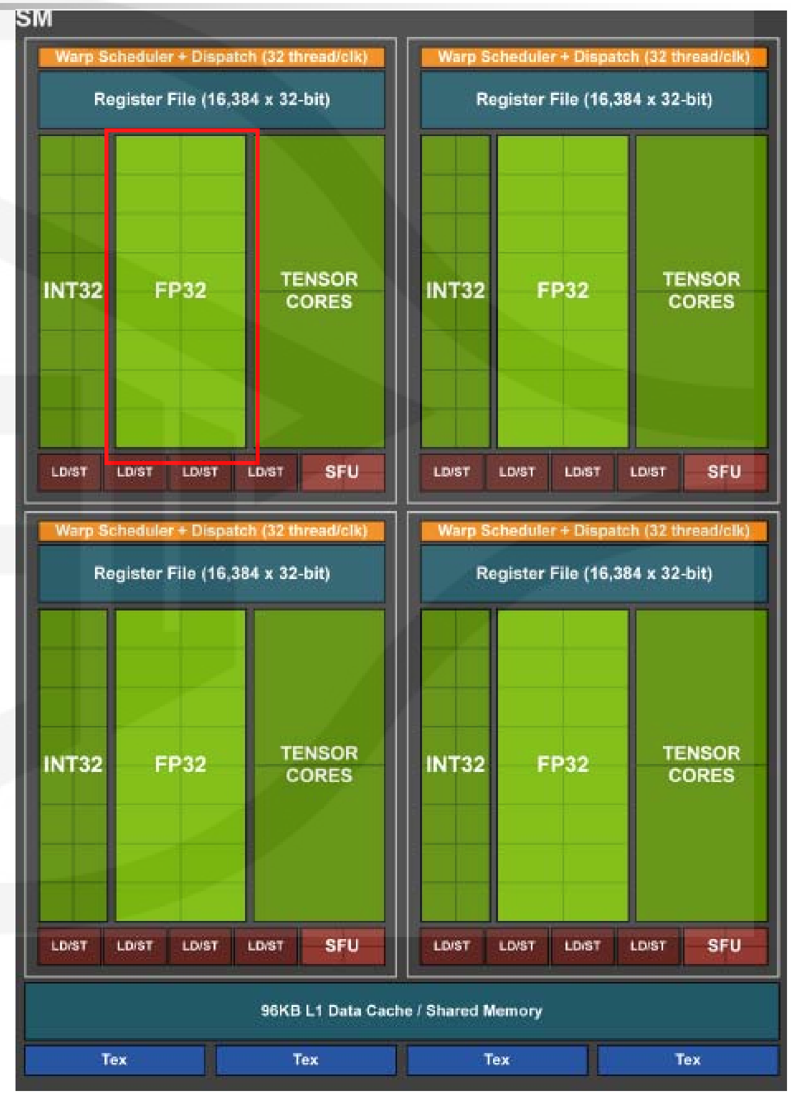
- Compare the outcome with vector_4

- Why notice data types are important ?
  (choose float64 or float32 ?)
- **Nvidia** has been dominating most of the market of scientific computing by GPU. Especially, in the deep learning. (Until the quantum computer replace them?)

GPU Architecture



last generation **Volta**



current **Turing**

Indexing and Slicing

```
vector = np.arange(10)
print(vector)
#indexing
print(vector[0])
print(vector[2])
print(vector[-3])
print(vector[:])
#indexing with stride
print(vector[::2])
print(vector[::-2])
#slice
print(vector[3:6])
print(vector[:6])
print(vector[6:])
#slice with stride
print(vector[:6:2])
print(vector[6::2])
```

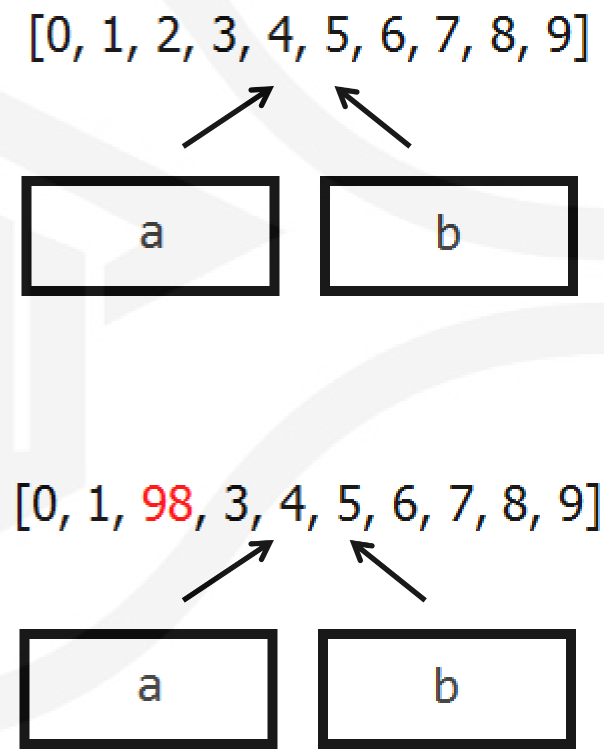
positive index

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
-10	*-9*	*-8*	*-7*	*-6*	*-5*	*-4*	*-3*	*-2*	*-1*

negative index

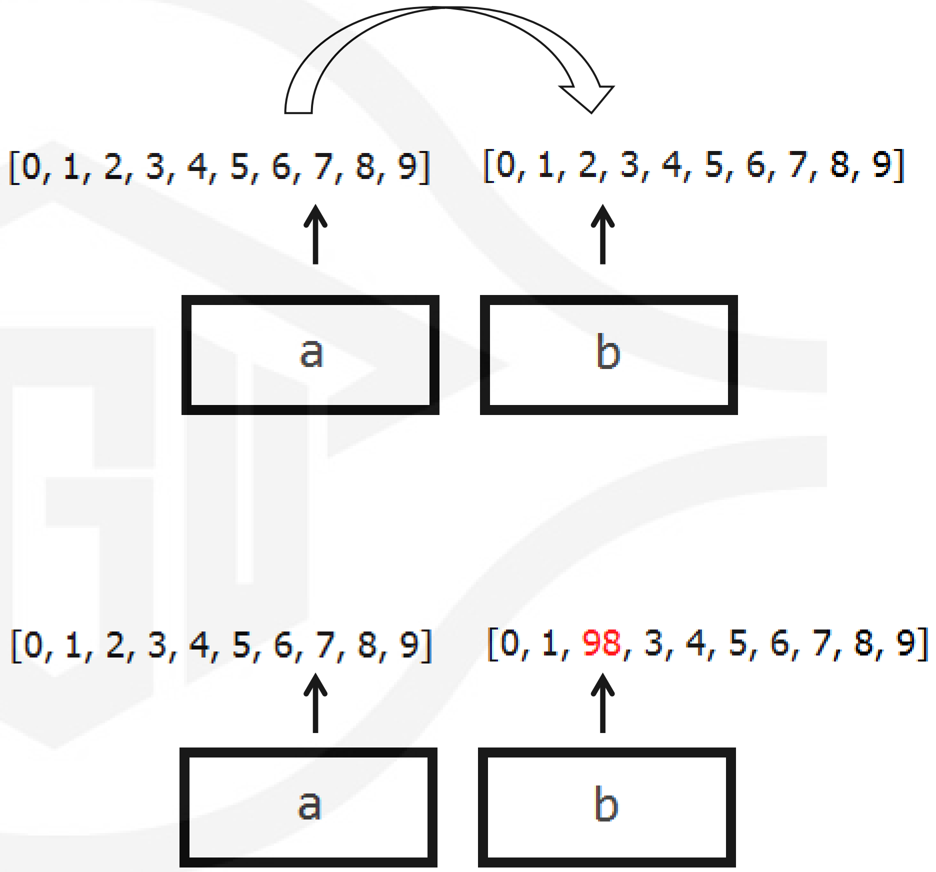
Assignment and Copy

```python
import numpy as np
a = np.arange(10)
b = a
b.itemset(2, 98)
if (a == b).all():
    print('equal')
else:
    print('not equal')
if a is b:
    print('same')
else:
    print('not same')
print(a)
print(b)
print(id(a))
print(id(b))
```



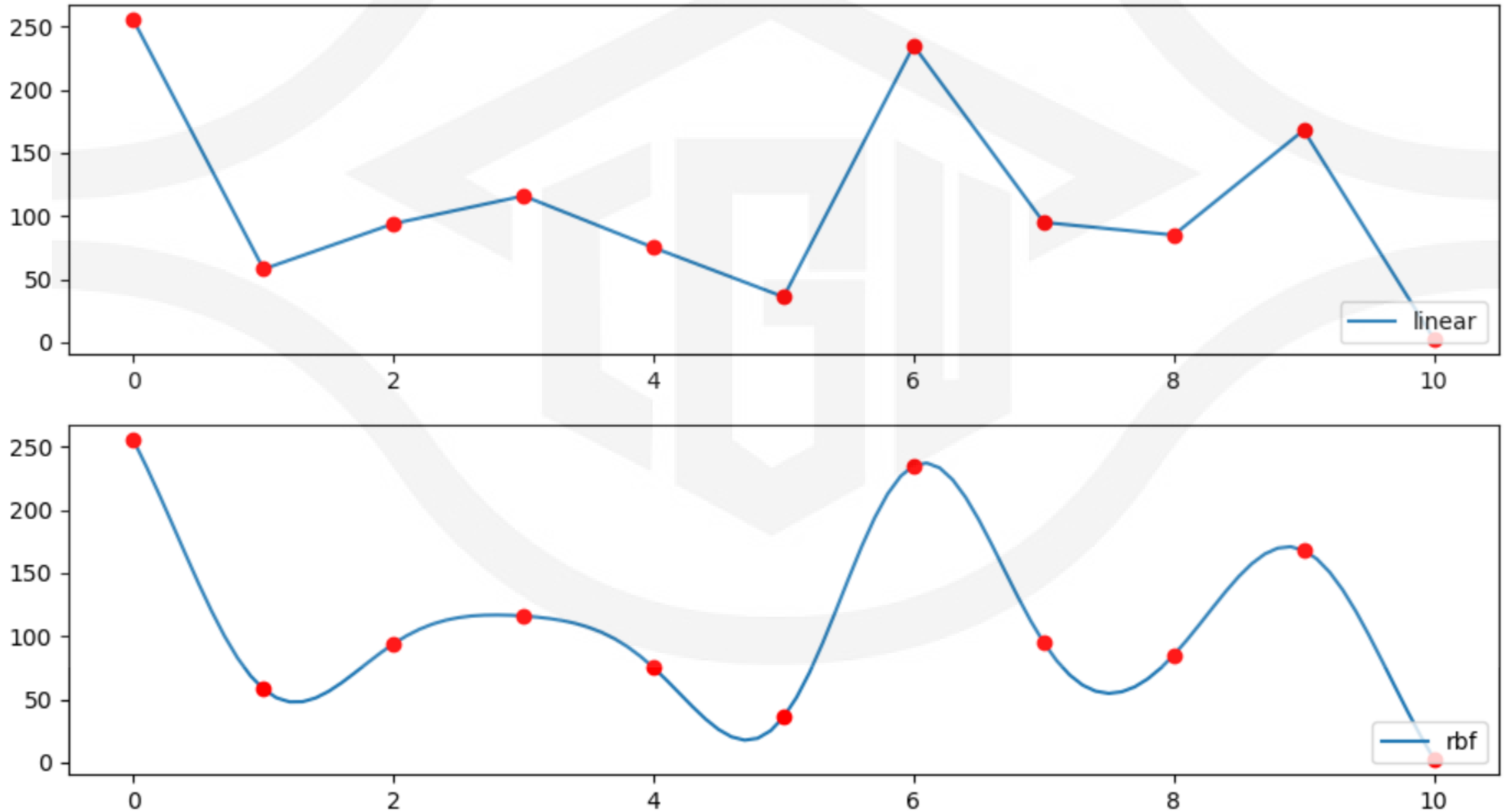
Assignment and Copy (cont.)

```
import numpy as np
a = np.arange(10)
b = a.copy()
b.itemset(2, 98)
if (a == b).all():
    print('equal')
else:
    print('not equal')
if a is b:
    print('same')
else:
    print('not same')
print(a)
print(b)
print(id(a))
print(id(b))
```



Visualization in Python

- Interpolate discrete points by the Gaussian kernel and linear method then plot it



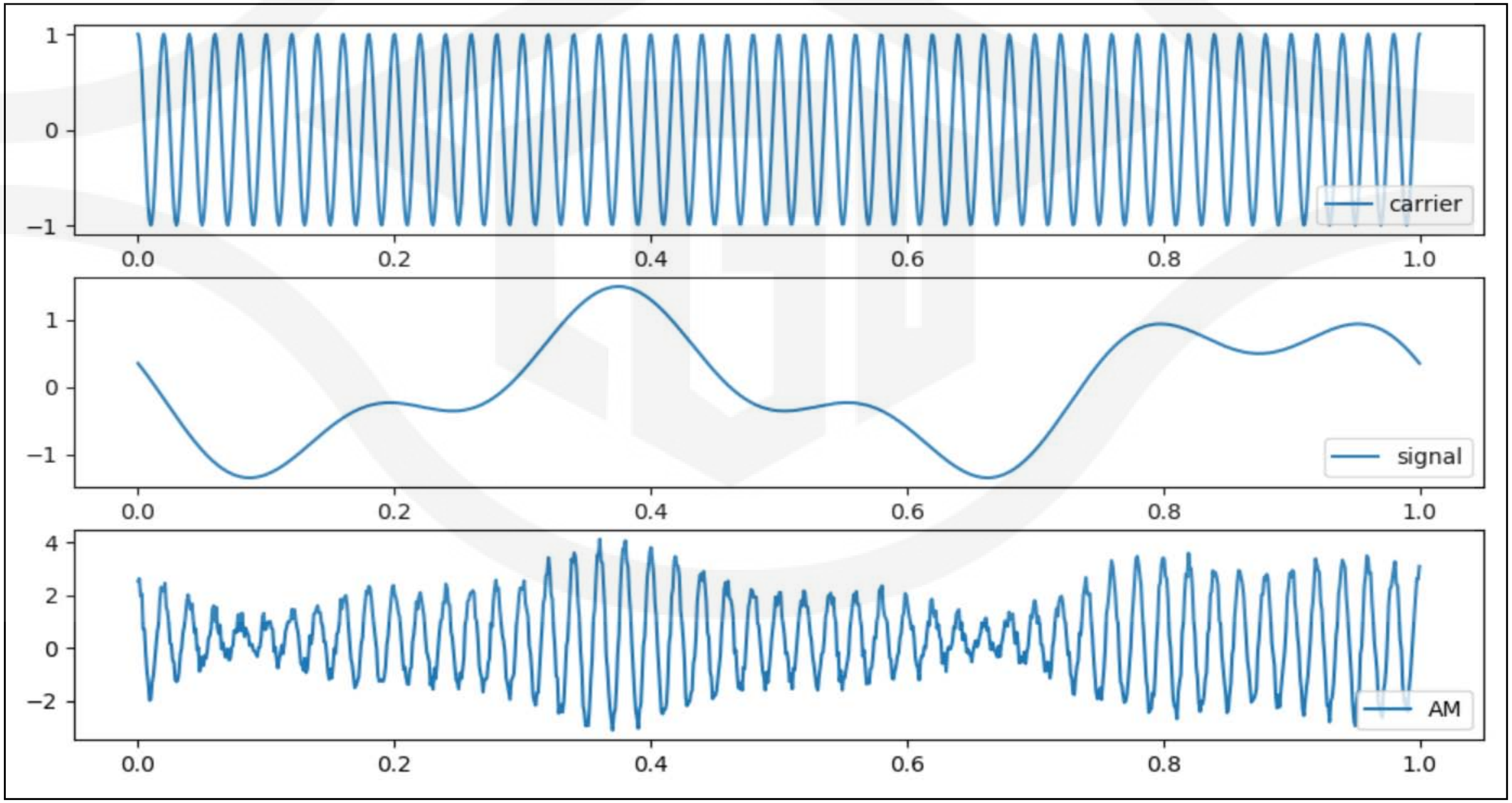
Using Matplotlib and Scipy

```
import matplotlib.pyplot as plt
import scipy.interpolate as spI
import numpy as np
x = np.linspace(0,10,11)
y = np.array([255,58,94,116,75,36,235,95,85,168,3])
xnew = np.linspace(0,10,101)
newfunc_l = spI.interp1d(x, y, kind='linear')
ynew_l = newfunc_l(xnew)
newfunc_g = spI.Rbf(x, y, kind='gaussian')
ynew_g = newfunc_g(xnew)
plt.subplot(211)
plt.plot(xnew,ynew_l,label=str('linear'))
plt.plot(x,y,"ro")
plt.legend(loc="lower right")
plt.subplot(212)
plt.plot(xnew,ynew_g,label=str('rbf'))
plt.plot(x,y,"ro")
plt.legend(loc="lower right")
plt.show()
```

Class Exercise

- AM is an old modulation method, please use the skills learned today to draw the following picture. Add noise to simulate the real situation.

$$\text{signal} = \cos\left(2\pi f_1 t + \frac{\pi}{2}\right) + \frac{1}{2}\cos\left(2\pi f_2 t + \frac{\pi}{4}\right), f_1 = 2, f_2 = 5 \qquad \text{carrier} = \cos(2\pi f t),\ f = 50$$



Exercise Hint

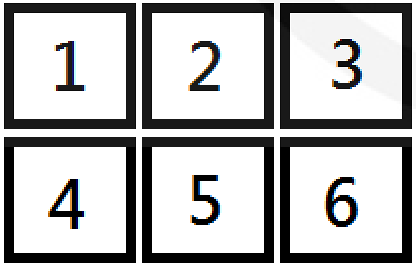
```
#import module

#frequency
f_c = 50 #50Hz
...
#time
t = np.linspace(0, 1, 1000)
#carrier
carrier = np.cos(2*np.pi*f_c*t)
#signal
signal =
#am = (signal+2)*carrier
am =
am = am + 0.8*np.random.rand(1000)
#plot
...
plt.show()
```

Create Matrix and Tensor

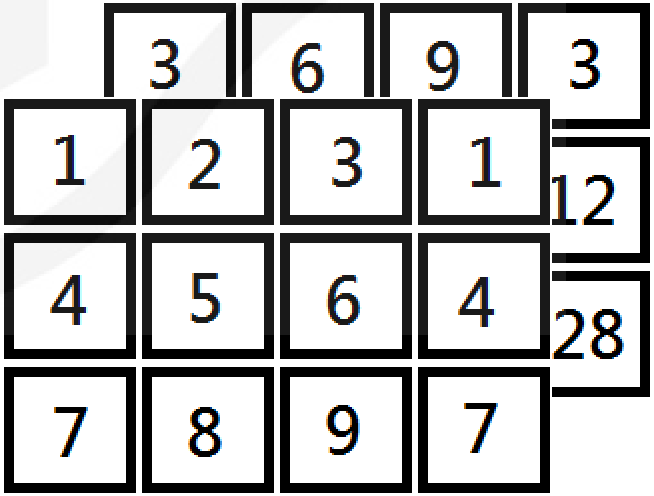
```
matrix_1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrix_1)
tensor_1 = np.array([[[1, 2, 3, 1], [4, 5, 6, 4],
[7, 8, 9, 7]],[[3, 6, 9, 3], [12, 15, 18, 12],
[28, 32, 36, 28]]])
print(tensor_1)
```

- check dimension (important in data processing)

```
print(matrix_1.shape)
print(tensor_1.shape)
```



matrix



tensor

Dimension Transformation

- Change the vector to a matrix (tensor) by dimension transformation and vice versa

```
vector = np.arange(10)
matrix_2 = vector.reshape(2, 5)
print(matrix_2)
print(vector)
matrix_3 = vector.resize(2, 5)
print(matrix_3)
print(vector)
```

similar but not identical

```
vector_r = matrix_2.reshape(matrix_2.shape[0]*matrix_2.shape[1])
print(vector_r)
vector_f = matrix_2.flatten()
print(vector_f)
```

Study Reshape and Resize in Depth

```
a = np.array([1, 2, 3,
4, 5, 6])
b = a
a.resize(2, 3)
print(a)
print(b)
if a is b:
    print('same')
else:
    print('not same')
print(id(a))
print(id(b))
```

```
a = np.array([1, 2, 3,
 4, 5, 6])
b = a.reshape(2, 3)
print(a)
print(b)
if a is b:
    print('same')
else:
    print('difference')
print(id(a))
print(id(b))
```

Indexing and Slicing at Matrix

- Pythonic coding style

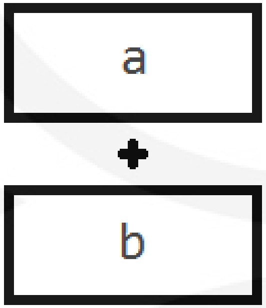
```
a = np.arange(16)
print(a)
b = a.reshape(4, 4)
print(b)
c = b[[1, 3], 2:]
print(c)
```

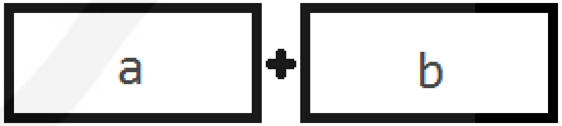
x[row, column]

```
d = np.reshape(np.arange(16), (4, 4))[[1, 3], 2:]
print(d)
```

Array Combination

```
a = np.random.rand(2, 3)
print(a)
b = np.random.rand(2, 3)
print(b)

c = np.concatenate((a, b),axis=0)
print(c)
print(c.shape)

d = np.concatenate((a, b),axis=1)
print(d)
print(d.shape)
```

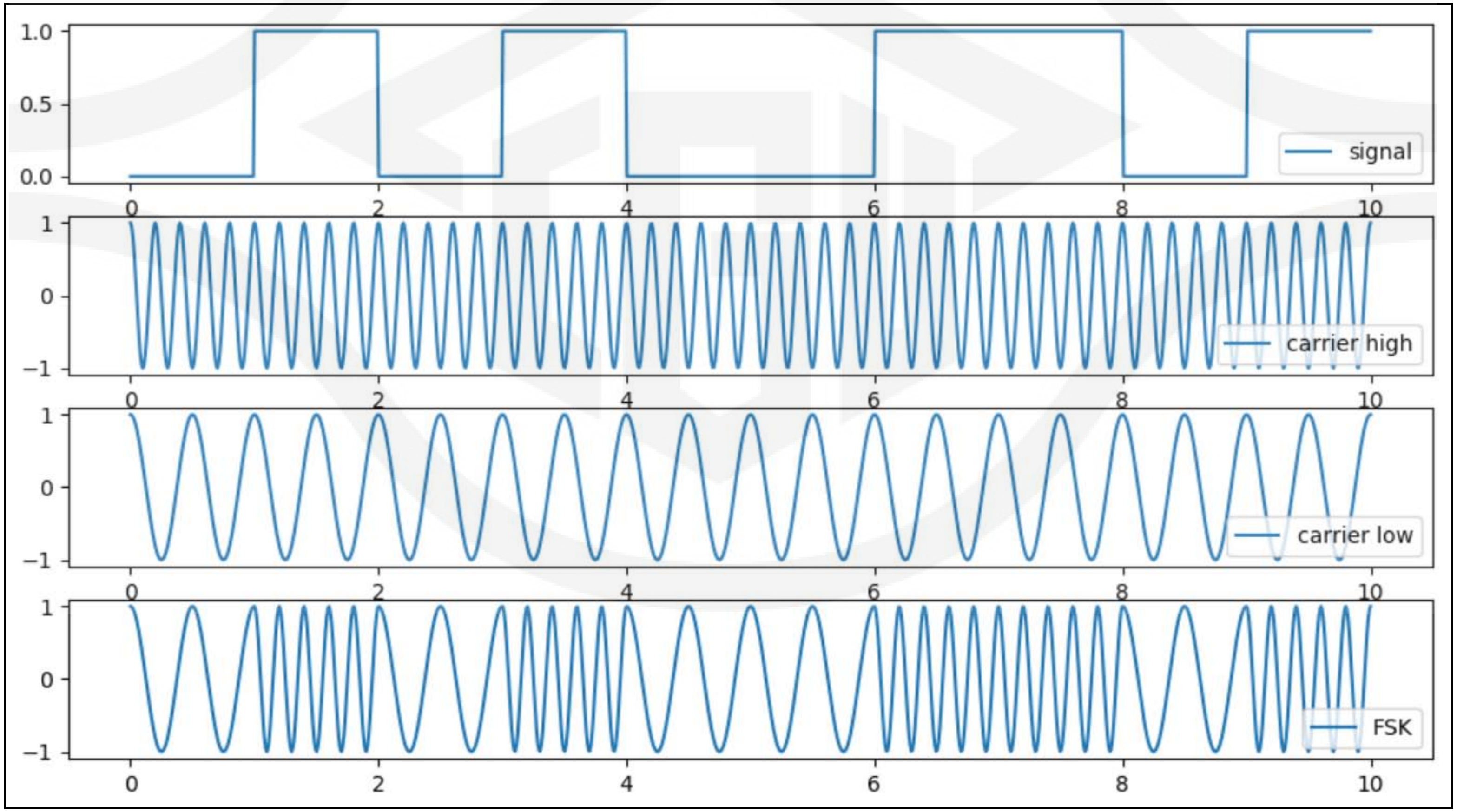





```
np.set_printoptions(precision=2)
```

Class Exercise

- FSK is a digital modulation technology that uses two different frequencies to encode the signal. Please draw the following picture.
- f_{high} = 5 , f_{low} = 2 , bit stream = 0101001101



Exercise Keynote

- Array creation
- Understanding the relationship between np.linspace and plot method
- Control flow (for..., if...)
- Array split and combination
- Using function (optional)