# Foundations of Machine Learning

*(Environment, Editor, Python, Numpy)*

Rowel Atienza, PhD

University of the Philippines

github.com/roatienza

2022

## Outline

Environment, Code Editor

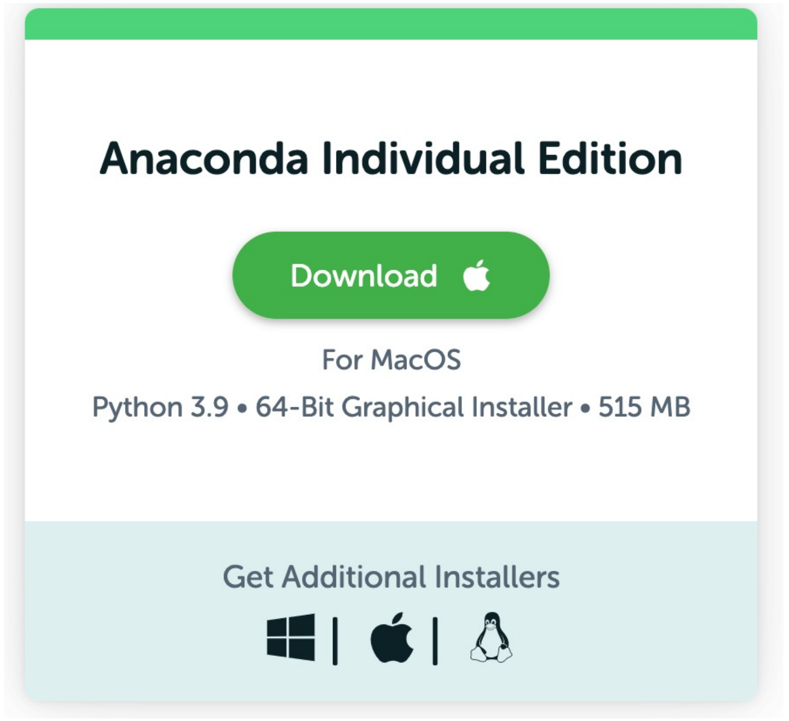
Python

Tensor library – numpy

GitHub

## Container Environment

### Anaconda



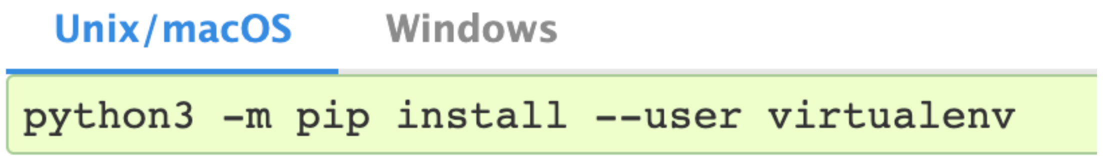
Anaconda Individual Edition

For MacOS

Python 3.9 • 64-Bit Graphical Installer • 515 MB

Get Additional Installers

### `venv`



```
python3 -m pip install --user virtualenv
```

## Container Environment

### Anaconda

```
conda create –name ml
```

### Venv

```
python3 –m venv ml
```

## Container Environment

### Anaconda

```
conda activate ml
```

### `Venv`

```
source ml/bin/activate
```

## Python package installer

`pip3 or pip`

Example:

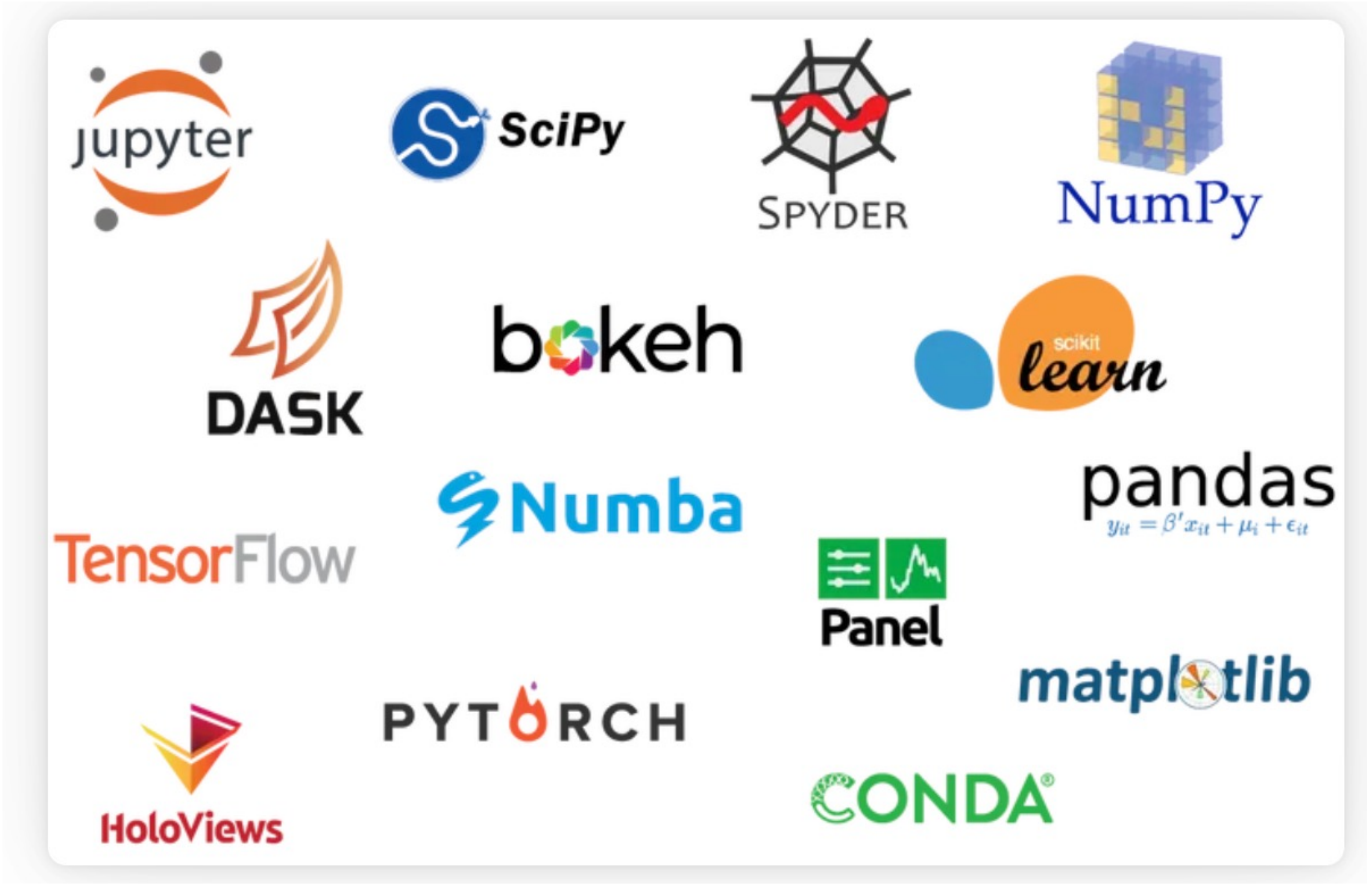
```
pip3 install einops
```

`conda`

Example:

```
conda install einops
```

## Anaconda – Machine Learning Toolkit



https://www.anaconda.com/

## `venv` – lightweight virtual environment

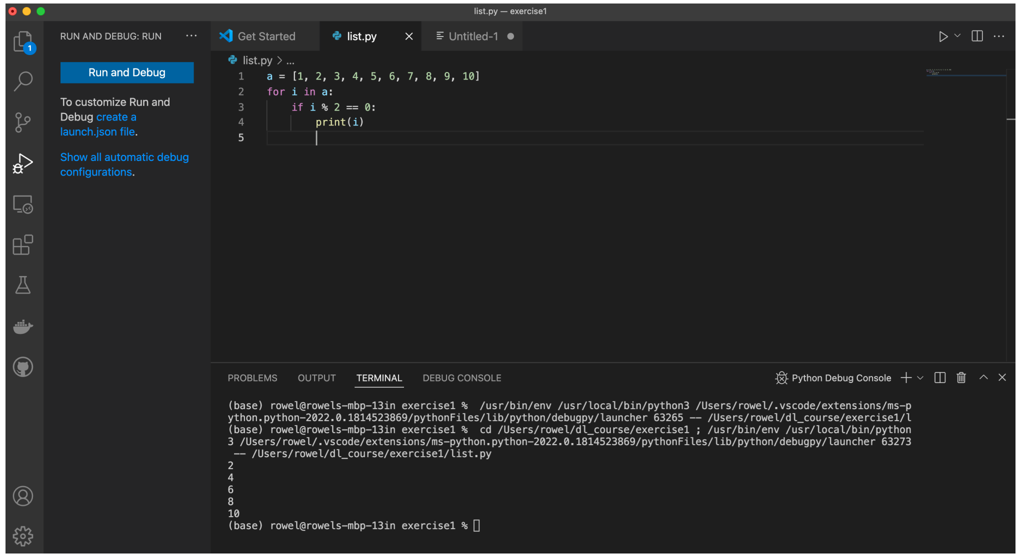
```
pip3 install virtualenv
python3 -m venv ml
source ml/bin/activate
which python3
/Users/rowel/ml/bin/python3
which pip3
/Users/rowel/ml/bin/pip3
```

## Code Editor

# Text Editor / IDE

## Visual Studio Code

Visual Studio Code



To customize Run and Debug create a launch.json file.

Show all automatic debug configurations.

```python
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for i in a:
    if i % 2 == 0:
        print(i)
```

```
(base) rowel@rowels-mbp-13in exercise1 %  /usr/bin/env /usr/local/bin/python3 /Users/rowel/.vscode/extensions/ms-p
ython.python-2022.0.1814523869/pythonFiles/lib/python/debugpy/launcher 63265 -- /Users/rowel/dl_course/exercise1/l
(base) rowel@rowels-mbp-13in exercise1 %  cd /Users/rowel/dl_course/exercise1 ; /usr/bin/env /usr/local/bin/python
3 /Users/rowel/.vscode/extensions/ms-python.python-2022.0.1814523869/pythonFiles/lib/python/debugpy/launcher 63273
 -- /Users/rowel/dl_course/exercise1/list.py
2
4
6
8
10
(base) rowel@rowels-mbp-13in exercise1 % 
```

✅ Recommended for its features

## vim

the editor

✅ Recommended for its portability

## Reference

- https://code.visualstudio.com/
- https://www.vim.org/
- https://www.anaconda.com/
- https://www.python.org/

# Python

https://github.com/dabeaz-course/practical-python

## Python

Scripting interpreted language

Exercise: activate python on your terminal

Exercise: create a new python source file in vscode

## Numbers

No need to declare the data type but common data types are supported : Boolean to complex numbers

Exercise:

- Generate 10 random integers. Store in a list. Print.
- Print the min and max
- Print in ascending order

Supports data type cast like in C

Exercise:

- Generate 10 random floats. Store in a list. Print.
- Convert all floats to int. Print.

## Strings

Declared using single or double quotes

```
name = "deep learning is fun"
```

Can be indexed

```
print(name[5:])
```

Can be concatenated

```
print(name + "!")
```

Supports string manipulation

```
print(name.replace("deep", "machine"))
```

Search

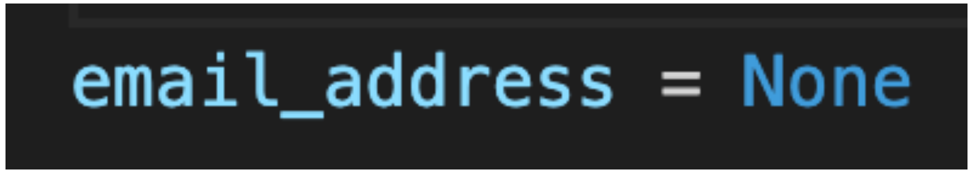
```
print("learn" in name)
```

String functions

```
print(name.upper())
```

## None

None is used as a placeholder for unsure or missing data type or value



```
email_address = None
```

# List

A **list** is a data structure that is a mutable, or changeable, ordered sequence of elements

Zero or more elements that are separated by commas

```python
x = [1, "fox", 3.4, [8, 16]]
```

Indexed

```python
print(x[1])
```

Concatenate

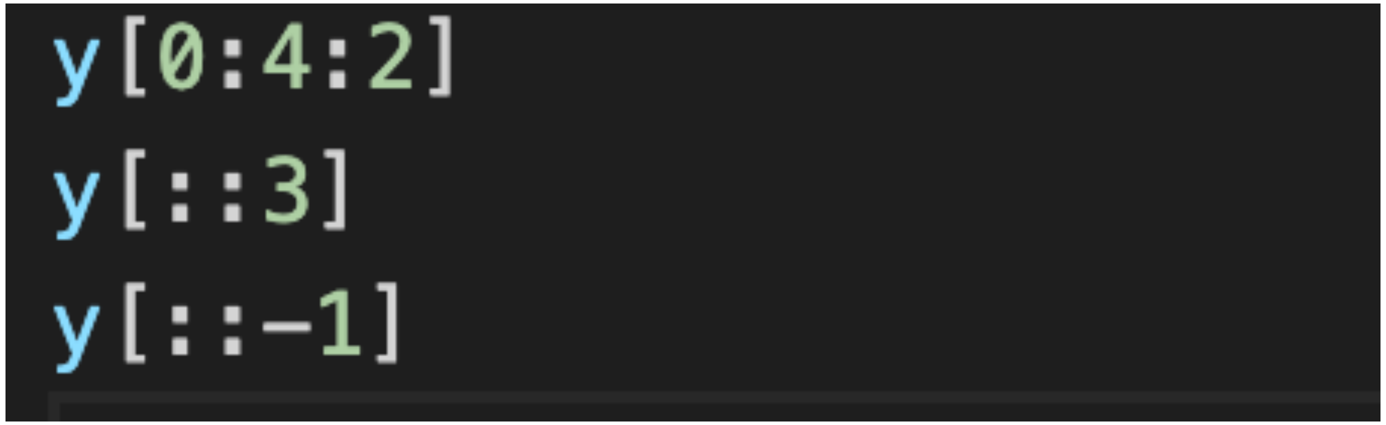
```python
y = [1, 2, 3, 4, 5]
z = [1, 4, 9, 16, 25, 36]
y + z
```

Append

```python
y.append(6)
```

## List - Slicing

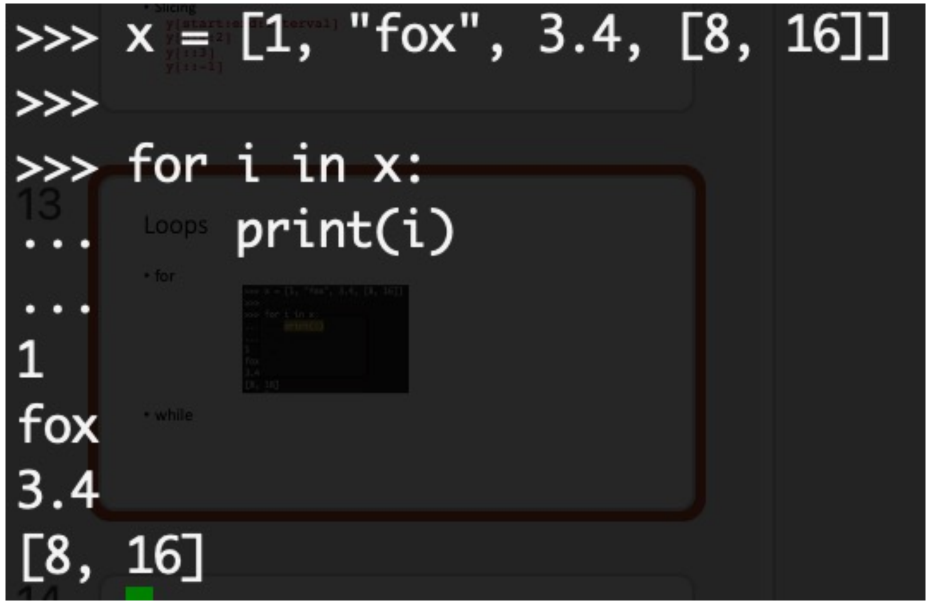
`y[start:end:interval]`



```
y[0:4:2]
y[::3]
y[::-1]
```

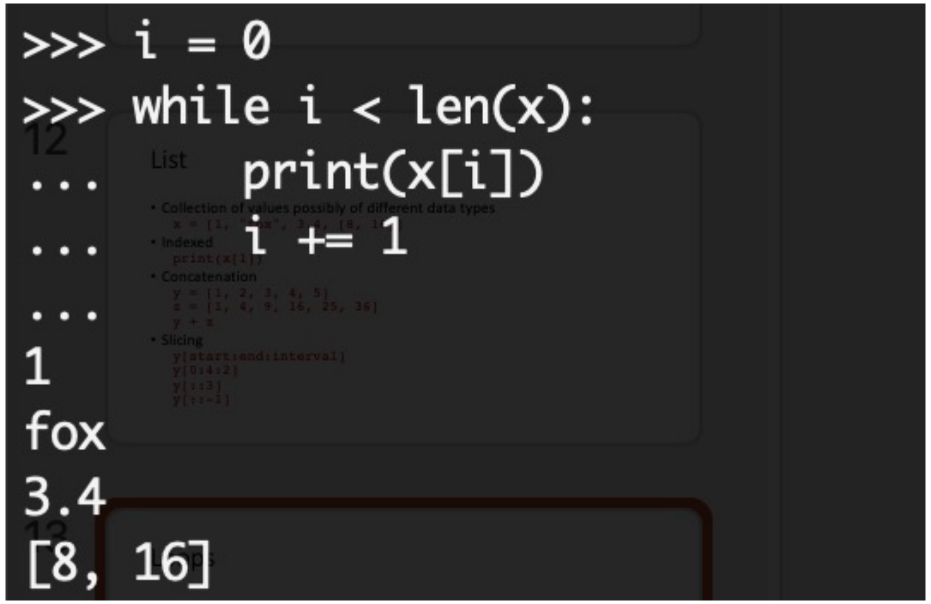
## Loops

- for



```
>>> x = [1, "fox", 3.4, [8, 16]]
>>>
>>> for i in x:
...     print(i)
...
1
fox
3.4
[8, 16]
```

- while



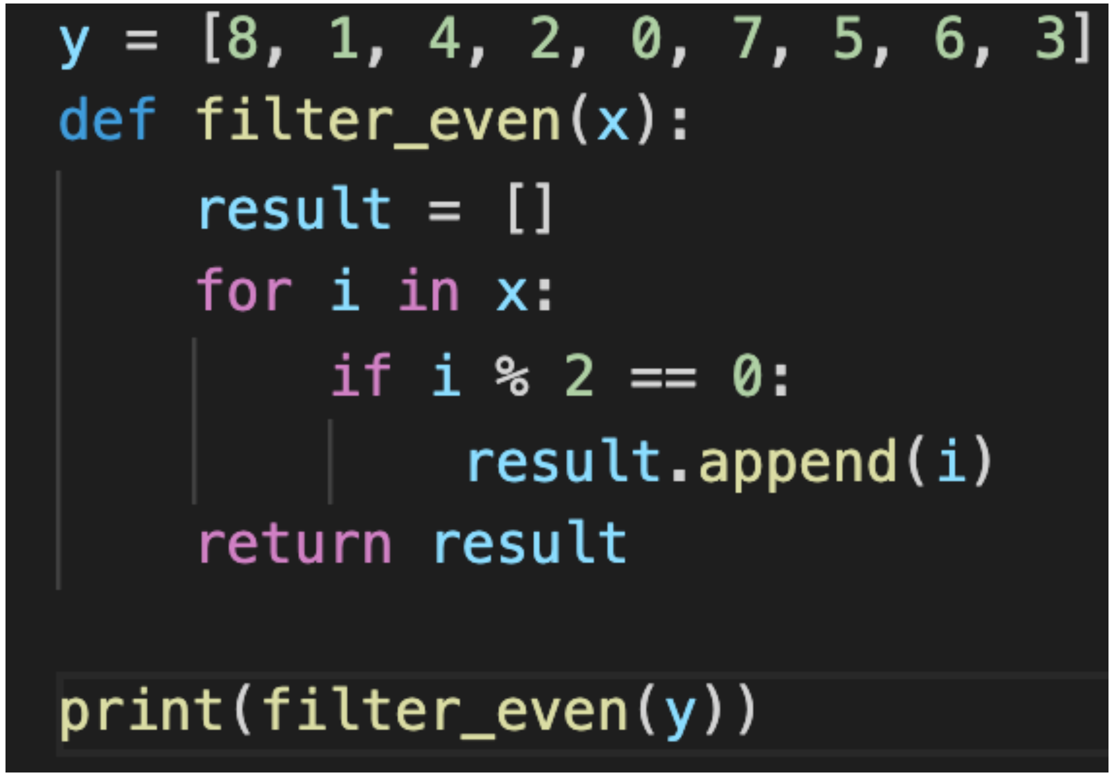
```
>>> i = 0
>>> while i < len(x):
...     print(x[i])
...     i += 1
...
1
fox
3.4
[8, 16]
```

## Function

We use the `def` keyword to define a function

A function has 0 or more inputs. Same with outputs.

Example: given a list of integers, get all even integers, store in a new list and print.



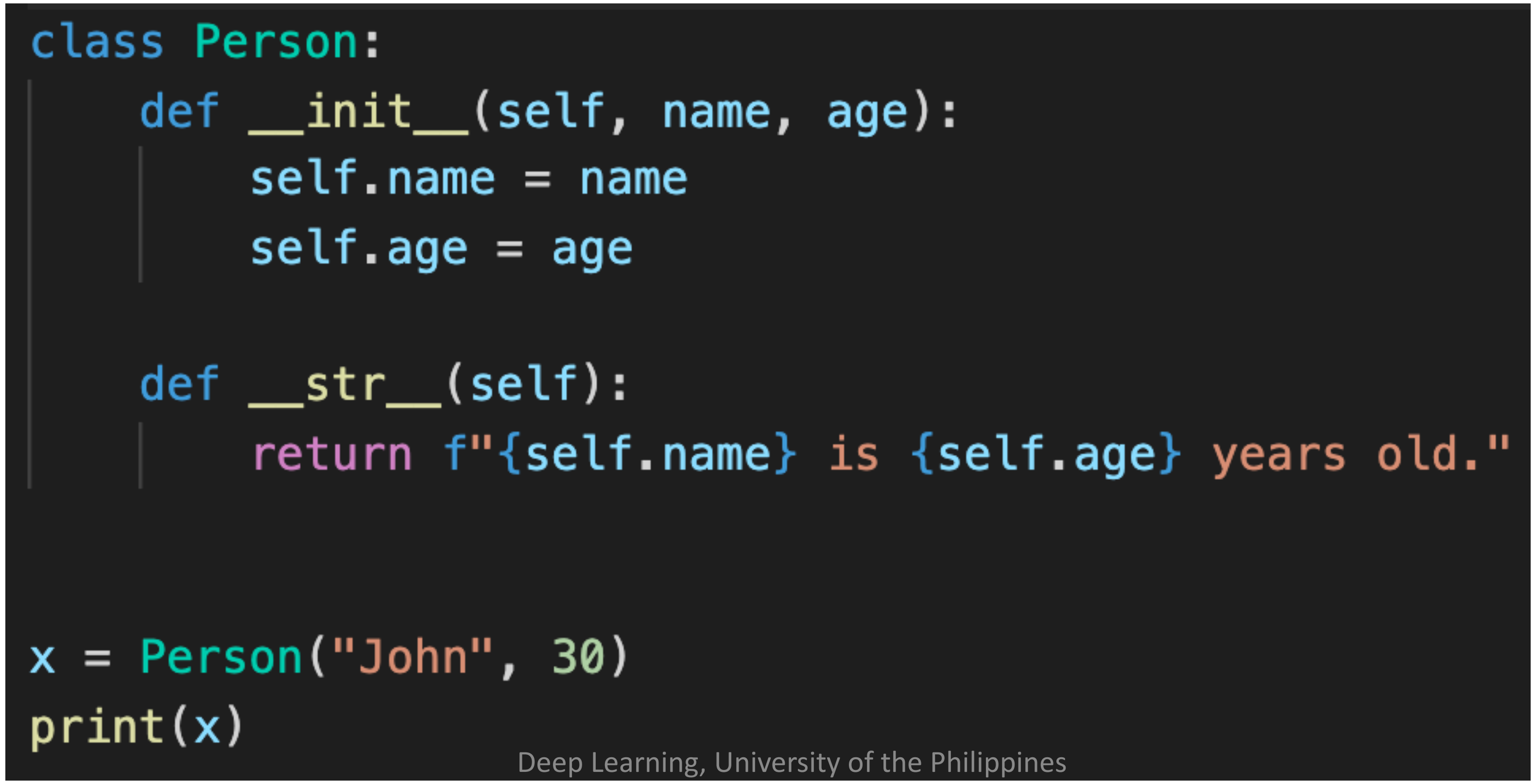
```python
y = [8, 1, 4, 2, 0, 7, 5, 6, 3]
def filter_even(x):
    result = []
    for i in x:
        if i % 2 == 0:
            result.append(i)
    return result

print(filter_even(y))
```

# Object Oriented

Class and inheritance

Methods and properties



```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"{self.name} is {self.age} years old."


x = Person("John", 30)
print(x)
```

## Reference

- Practical python https://github.com/dabeaz-course/practical-python

## Numpy

https://numpy.org/

## List vs Tuple or `a=[1, 2.2, ”the”]` vs `a=(1, 2.2, ”the”)`

| | `List` | `Tuple` |
|---|---|---|
| Mutable | Yes | No |
| Supported | `index, count` | `index, count` |
| Supported | `insert, append, pop, clear, remove, reverse` | |
| Use | Elements might change | Fixed elements |

## Numpy - Basics

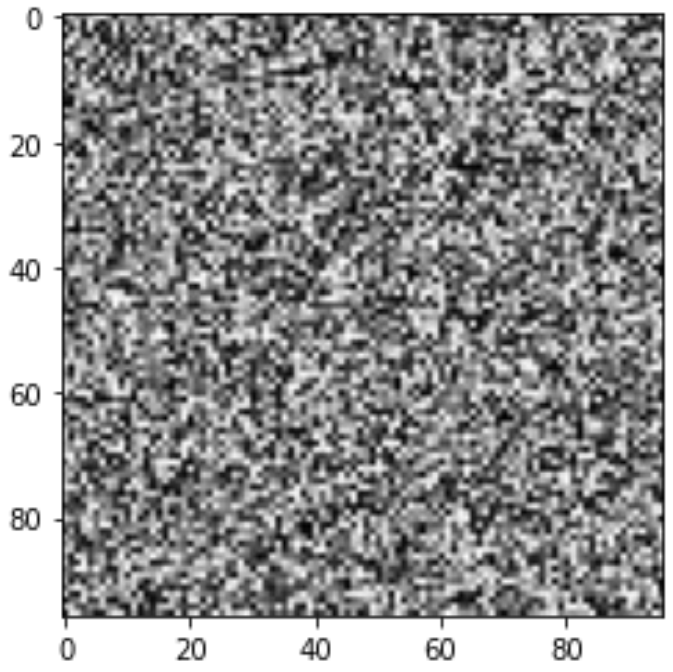
```
# Create an array
import numpy as np
a = np.array([[1,2,3], [4,5,6]])
# Data type: dtype('int64')
a.dtype
# Shape: (2, 3)
a.shape
# Number of dimensions: 2
a.ndim
```

## Numpy - Basics

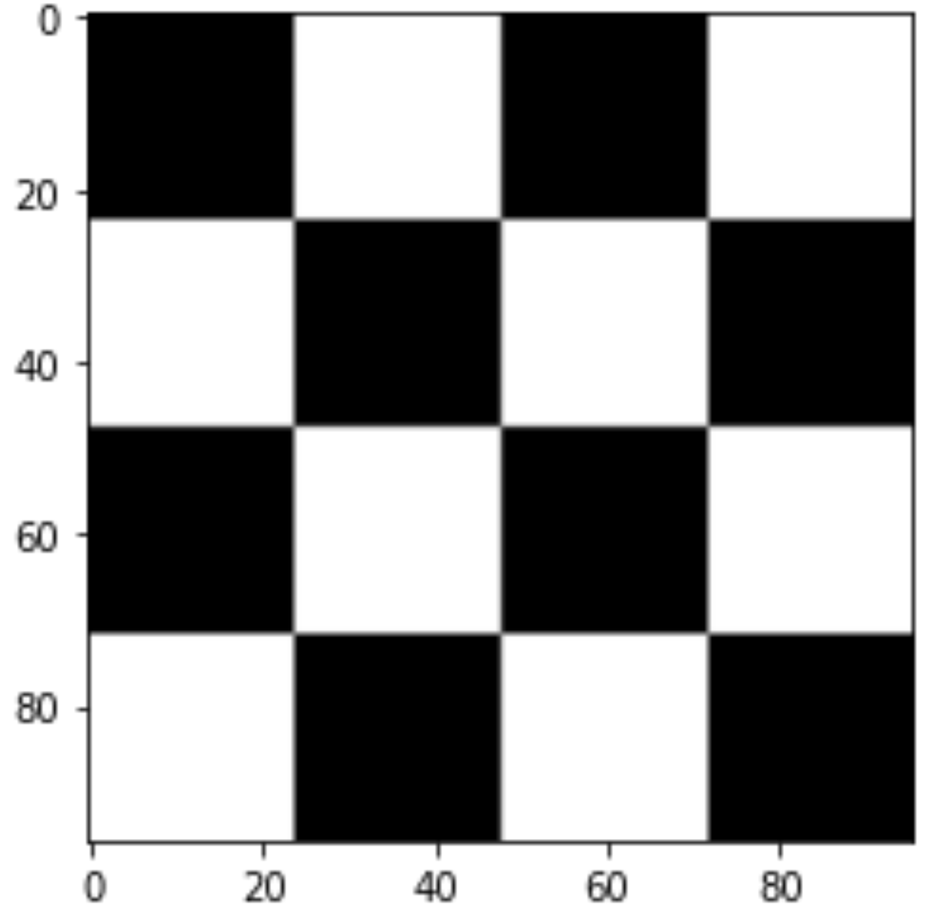
```
# Add a constant
2 + a
# Add 2 arrays
b = np.ones(a.shape)
a+b
# Multiply 2 arrays
a*b
# Matrix multiply 2 arrays
np.matmul(a,np.transpose(b))
a@np.transpose(b)
```

# Numpy for Data

```
img = np.random.randint(0,255,size=(96,96),dtype=np.uint8)
plt.imshow(img, cmap='gray', vmin=0, vmax=255)
plt.show()
```



# Chessboard Pattern



## Chessboard Pattern

```
img = np.one[illegible]96), dtyp[illegible]int8)*255
for i in range[illegible]
  img[i*24:(i+1)[illegible]1)*24] = 0

for i in range(2,4[illegible]
  img[i*24:(i+1[illegible](i[illegible](i-1)*24] = 0

for i in ra[illegible]):
  img[i*24:(i[illegible])*24, (i+2)*24[illegible]+3)*24] = 0
```

# Chessboard Pattern

```
def chessboard(shape):
  return np.indices(shape).sum(axis=0) % 2

img = chessboard((4,4))*255

img = np.repeat(img, (24), axis=0)
img = np.repeat(img, (24), axis=1)
```

# np.indices((4,4))

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

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

## `np.indices((4,4)).sum(axis=0)`

```
[[0 1 2 3]
 [1 2 3 4]
 [2 3 4 5]
 [3 4 5 6]]
```

## `np.indices((4,4)).sum(axis=0)%2`

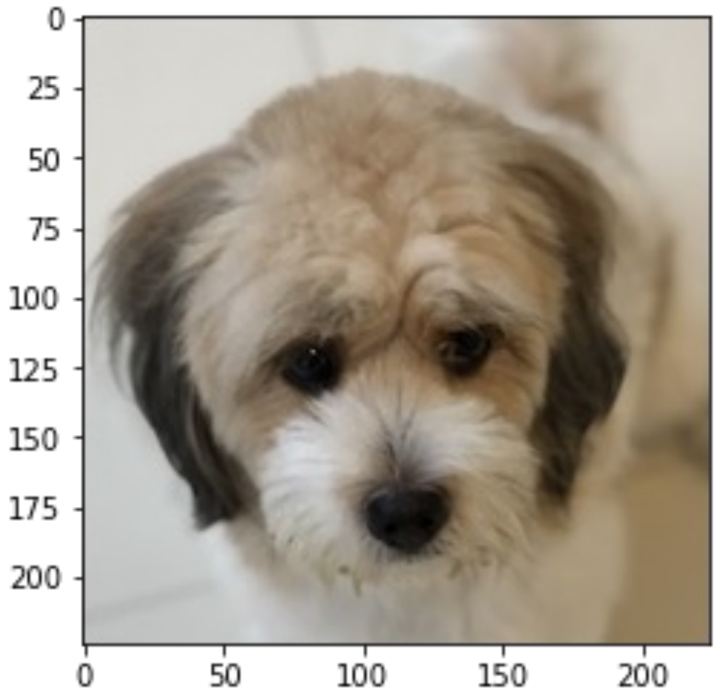
```
[[0 1 0 1]
 [1 0 1 0]
 [0 1 0 1]
 [1 0 1 0]]
```

Exercise:

Without using loops, find another algorithm that can generate this pattern.

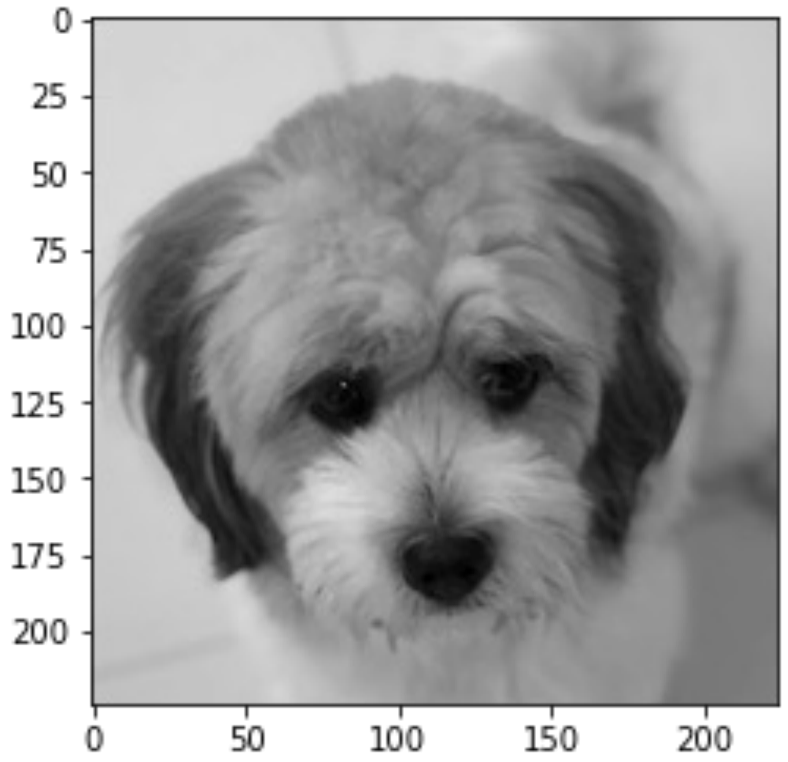
## Loading an image

```
from matplotlib import image
img = image.imread("aki_dog.jpg")
```



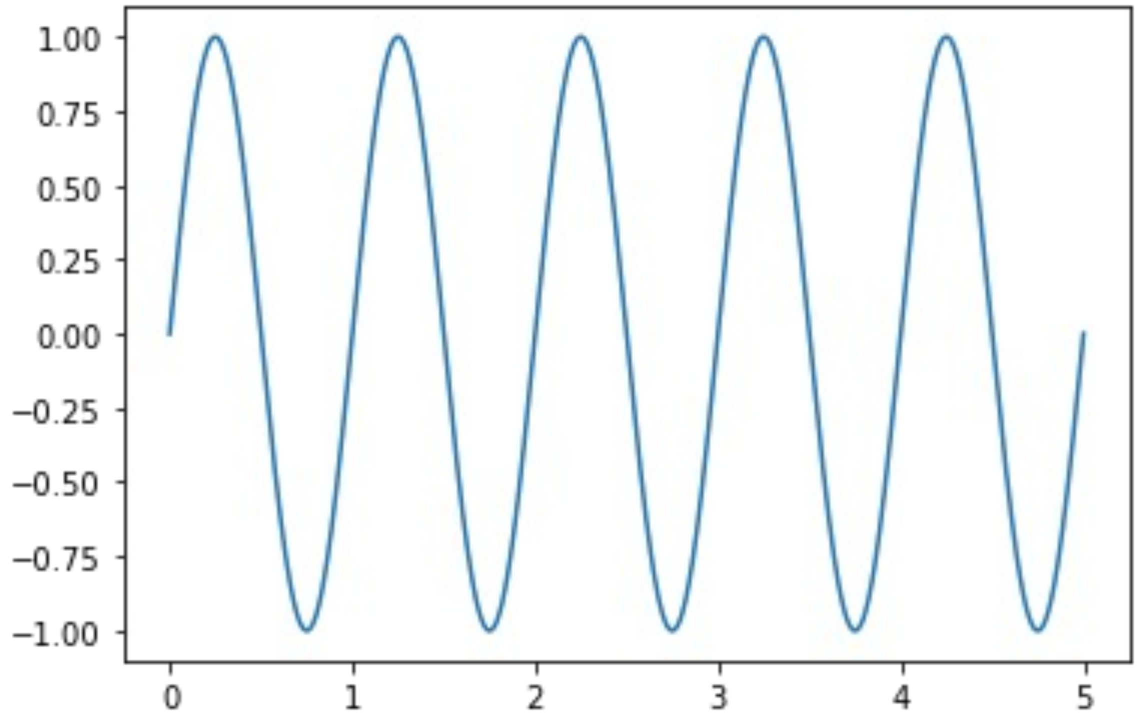
# RGB to Grayscale

```
img = np.mean(img, axis=-1)
```



## Synthetic Audio Waveform

```
samples_per_sec = 22050
freq = 1
n_points = samples_per_sec*5
t = np.linspace(0,5,n_points)
data = np.sin(2*np.pi*freq*t)
```



## Limitations of Numpy

Not designed for GPU execution

- Alternative: `cupy`

Different methods/APIs for different tensor operations

Many steps for complex linear algebra operations

- Alternative: `einsum` and `einops`

## Github

## Basic commands

Get the ml code:

```
    git clone https://github.com/roatienza/ml.git
```

Update:

```
    git pull
```

You can fork and add your own code:

```
    gid add my.py
    git commit –a –m “my own file”
    git push origin
```

# End