LIVE

EBU7240

Computer Vision

- labs and coursework-

Semester 1, 2021

Changjae Oh

Contents

- **Setting up your playground**
- Python / examples
- OpenCV / examples

Notes

- Following slides are guidelines **for those who start Python from scratch**

- You can use your own way to setup the environment
- **BUT** make sure that your code is runnable in the provided environment:
  - Python 3.8
  - *pyqt5*
  - *numpy*
  - *opencv-python*
  - *opencv-contrib-python*
  - *matplotlib*

Anaconda

- **A free and open-source distribution of the Python languages**
  - for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.),
  - aims to simplify package management and deployment.

ANACONDA®

Miniconda (Anaconda)

- **A free minimal installer for conda.**
  - A small, bootstrap version of Anaconda that includes only conda, Python, the packages they depend on, and a small number of other useful packages, including pip, zlib and a few others.
  - https://docs.conda.io/en/latest/miniconda.html

Windows installers

Windows

Python version	Name	Size	SHA256 hash
Python 3.9	Miniconda3 Windows 64-bit	58.1 MiB	b33797064593ab2229a0135dc69001bea05cb56a20c2f243b1231213642e260a
Python 3.8	Miniconda3 Windows 64-bit	57.3 MiB	8940cdd621557bc55743d6bb4518c6d343a4587127e76de808fb07e51df03fea
Python 3.7	Miniconda3 Windows 64-bit	55.8 MiB	9c031506bfcb0428a0ac46c9152f9bdd48d5bdaa83046691bf8e0a4480663c05
Python 3.9	Miniconda3 Windows 32-bit	55.3 MiB	24f438e57ff2ef1ce1e93050d4e9d13f5050955f759f448d84a4018d3cd12d6b
Python 3.8	Miniconda3 Windows 32-bit	54.5 MiB	f81c165384c18d1986e2ba2f86cef384bc62266c46b34cd3d274e751ff5d91ed
Python 3.7	Miniconda3 Windows 32-bit	55.3 MiB	a1bb8338be12ee09dbd4cab9dcc2fbdc99f65d99281dd2c07d24ad0f23dd1f7c

Download → Install

PyCharm

- **Making your Python coding easier**

PyCharm

- **Download -> Install!**

Version: 2021.2.1
Build: 212.5080.64
27 August 2021

System requirements

Installation Instructions

Download PyCharm

Windows macOS Linux

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Download

Free trial

Community

For pure Python development

Download

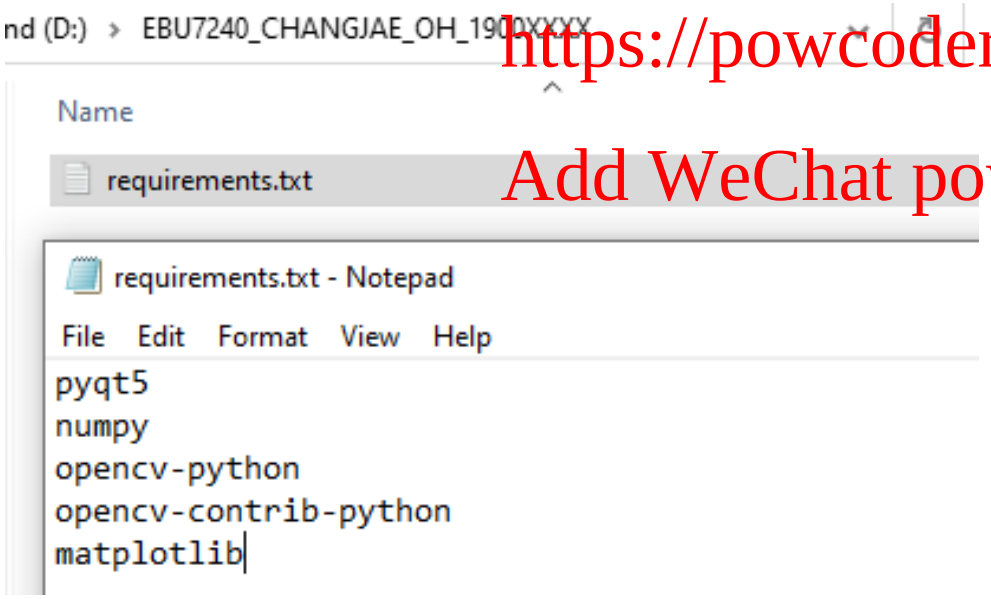
Free, built on open-source

Setup

Setup

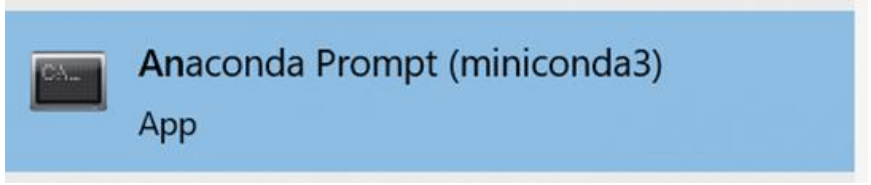
1. Your directory

2. Put "requirements.txt" (in CMPlus) in the above folder



Setup

3. Run Anaconda Prompt



4. Create the virtual environment (your playground)


```
(base) D:\EBU7240_CHANGJAE_OH_1900XXXX>conda create -n ebu7240_CV python=3.8
```


5. Activate the environment


```
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate ebu7240_CV
#
# To deactivate an active environment, use
#
#     $ conda deactivate


(base) D:\EBU7240_CHANGJAE_OH_1900XXXX>conda activate ebu7240_CV

(ebu7240_CV) D:\EBU7240_CHANGJAE_OH_1900XXXX>
```

Setup

6. Installing packages from "requirements.txt"

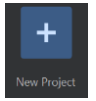
```
(base) D:\EBU7240_CHANGJAE_OH_1900XXXX>conda activate ebu7240_CV

(ebu7240_CV) D:\EBU7240_CHANGJAE_OH_1900XXXX>pip install -r requirements.txt
```


Your virtual environment includes all libraries you need for this module

Now, let's connect this virtual environment with PyCharm

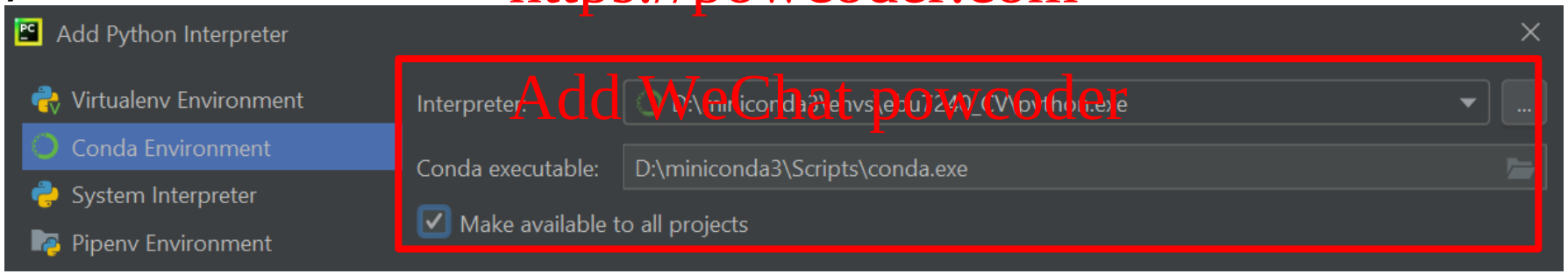
Setup

9. Open PyCharm -> Projects -> New Project 

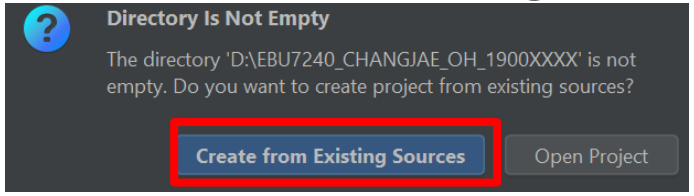
10. Set location as your folder 

11. Previously configured interpreter → click "…"

12. Load your conda environment

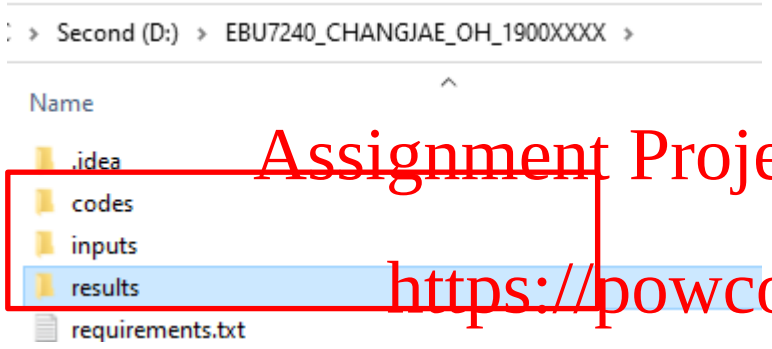


13. Click "OK" → "Create" → "Create from Existing Sources"

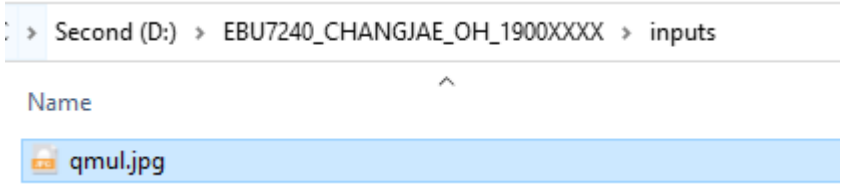


Setup

13. Create the subfolders you will need

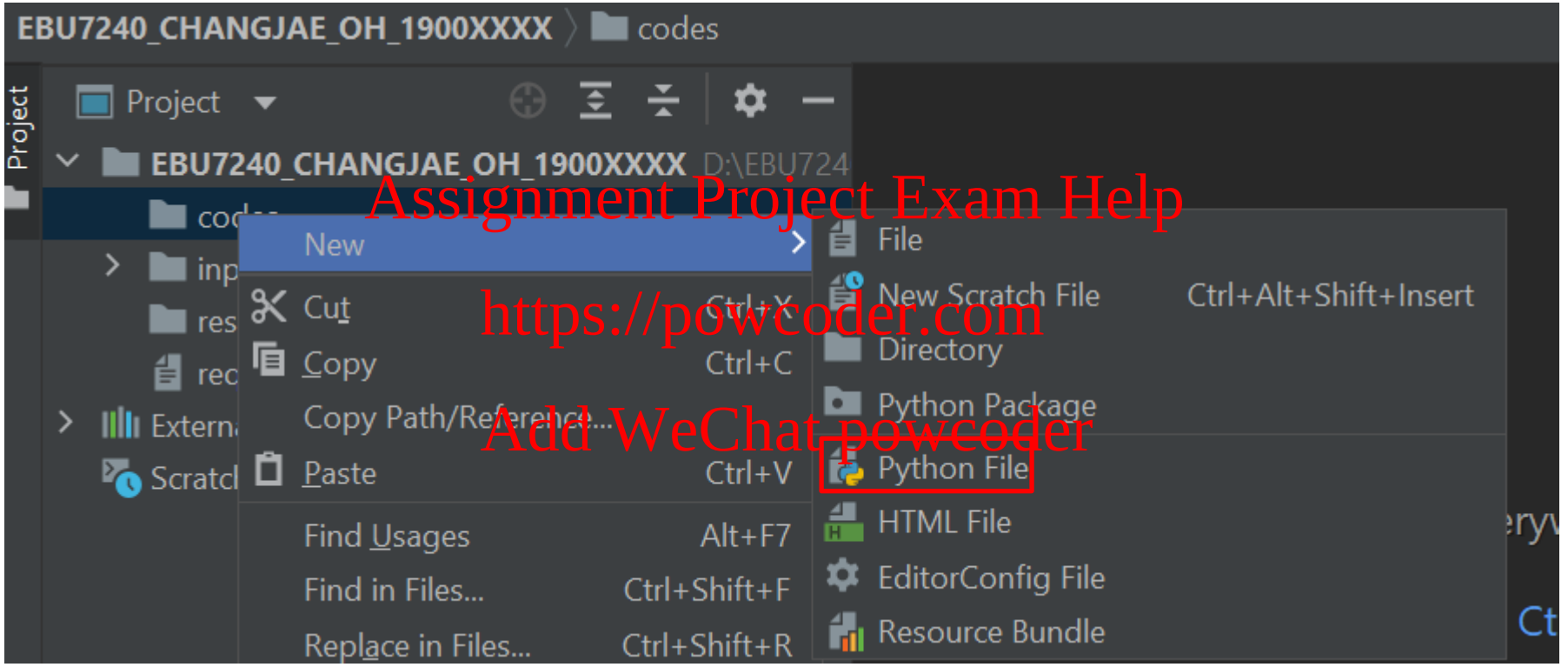


14. Put any image in ./inputs/



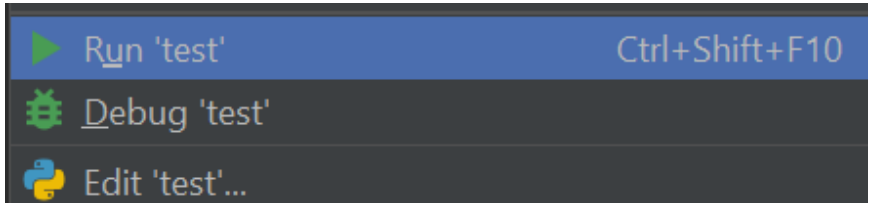
Setup

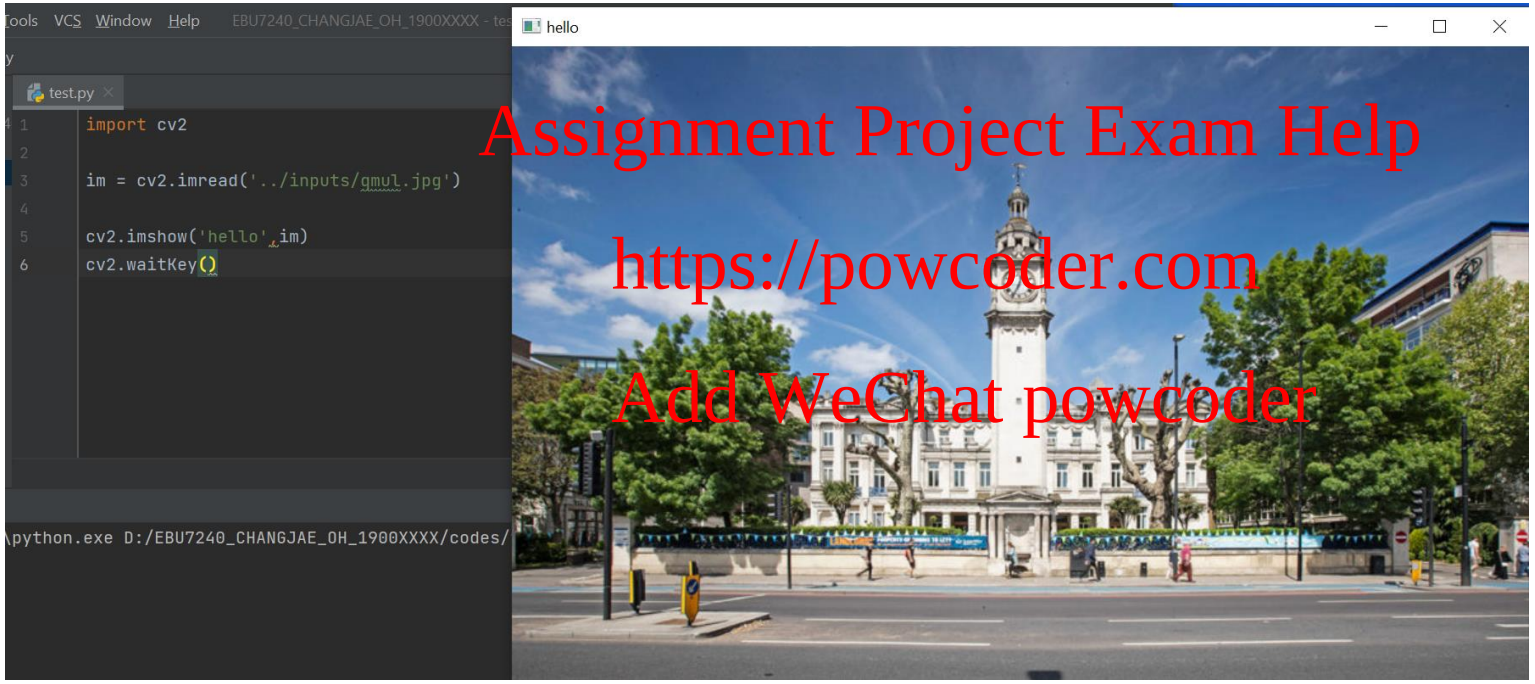
15. Create python File in the ./codes/



Setup

16. Right click -> Run



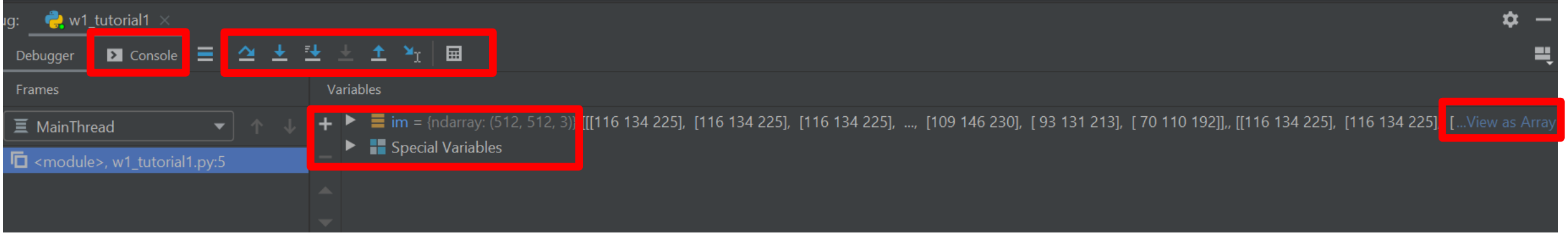


Now you are ready to code!

Useful tips

- Debug

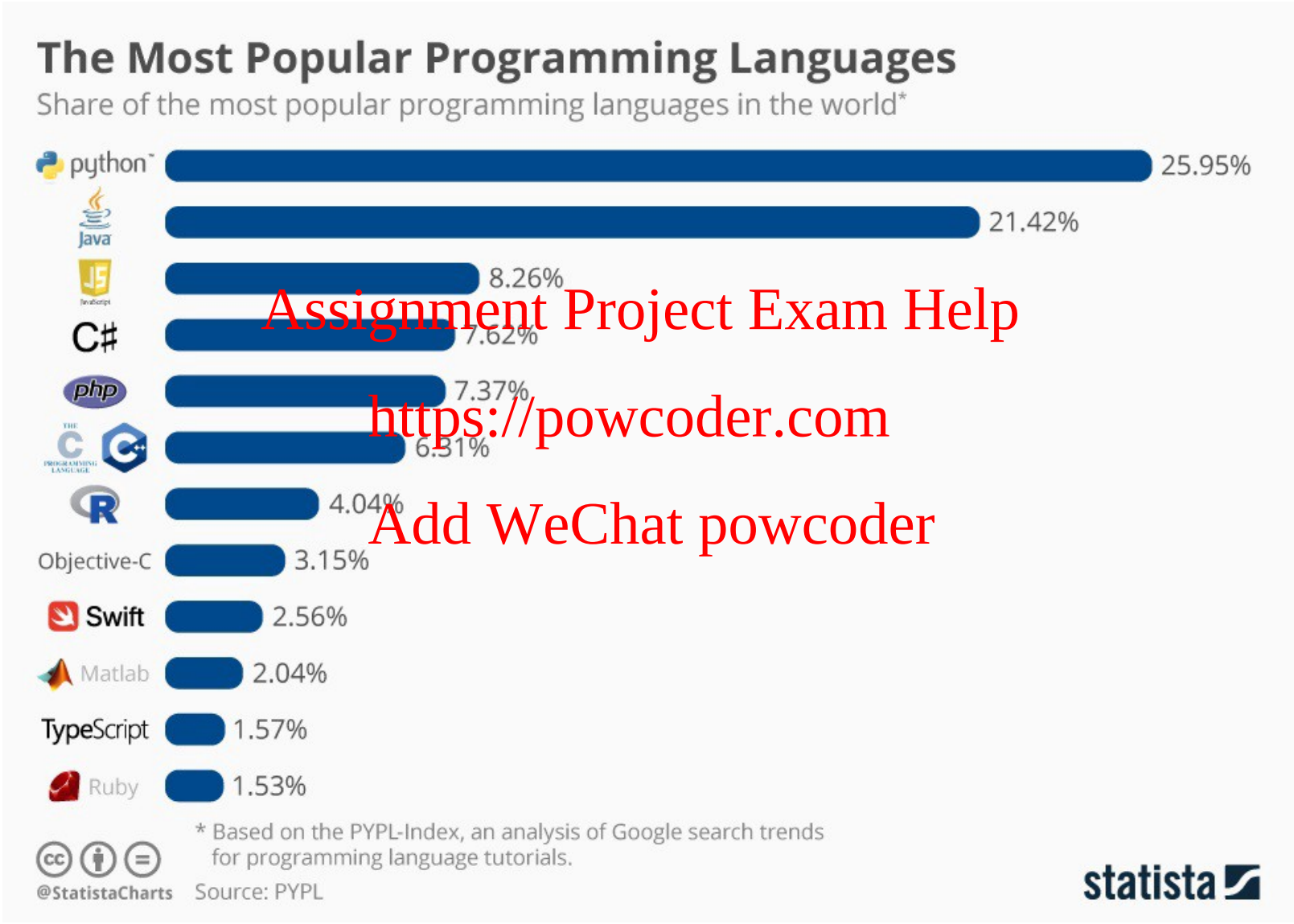
```
1  ## image read, show, write
2  import cv2
3
4  im = cv2.imread('./inputs/lena.jpg')  im: [[[116 134 225],  [116 134 225],  [116 134 225],  ...,  [109 146 230],  [ 93 131 213],  [ 70 110 192]],
5  im_gray1 = cv2.imread('./inputs/lena.jpg', 0)
6  im_gray2 = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
7
8  cv2.imshow('lena', im)
9  cv2.waitKey()
10
11 cv2.imshow('lena_gray', im_gray1)
12 cv2.waitKey()
13
14 cv2.imshow('lena_gray', im_gray2)
15 cv2.waitKey()
```

Contents

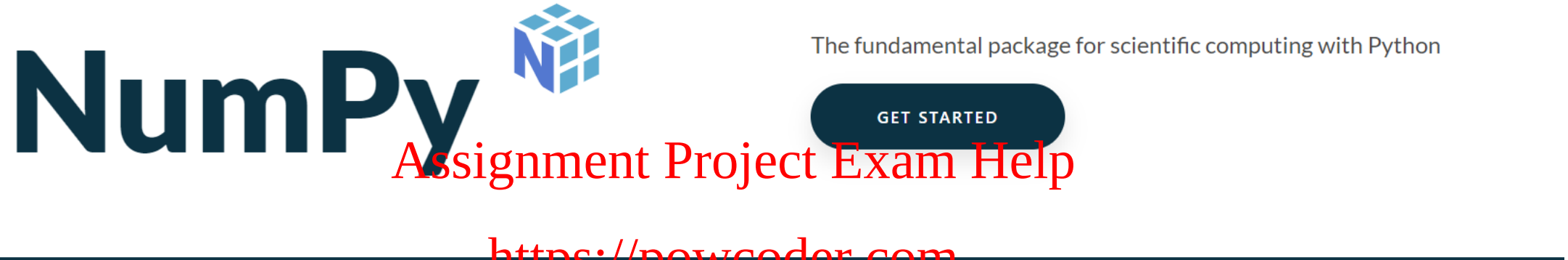
- Setting up your playground
- **Python**
- OpenCV

Why Python?



https://www.upgrad.com/blog/reasons-why-python-popular-with-developers/

NumPy



Python (numpy) vs MATLAB

Python

```
>>> import numpy as np

# Create row vector
>>> row = np.array([1, 2, 3])
>>> row
array([1, 2, 3])

# Transpose
>>> col = row.T

# Compute inner product
>>> inner = np.dot(row,col)
>>> inner
14

# Compute outer product
>>> outer = np.dot(col,row)
>>> outer
14
```

MATLAB

```
% Create row vector
>> row = [1 2 3]
row =
     1     2     3

% Transpose
>> col = row';

% Compute inner product
>> inner = row*col
inner =
    14

% Compute outer product
>> outer = col*row
outer =
     1     2     3
     2     4     6
     3     6     9
```

More Info

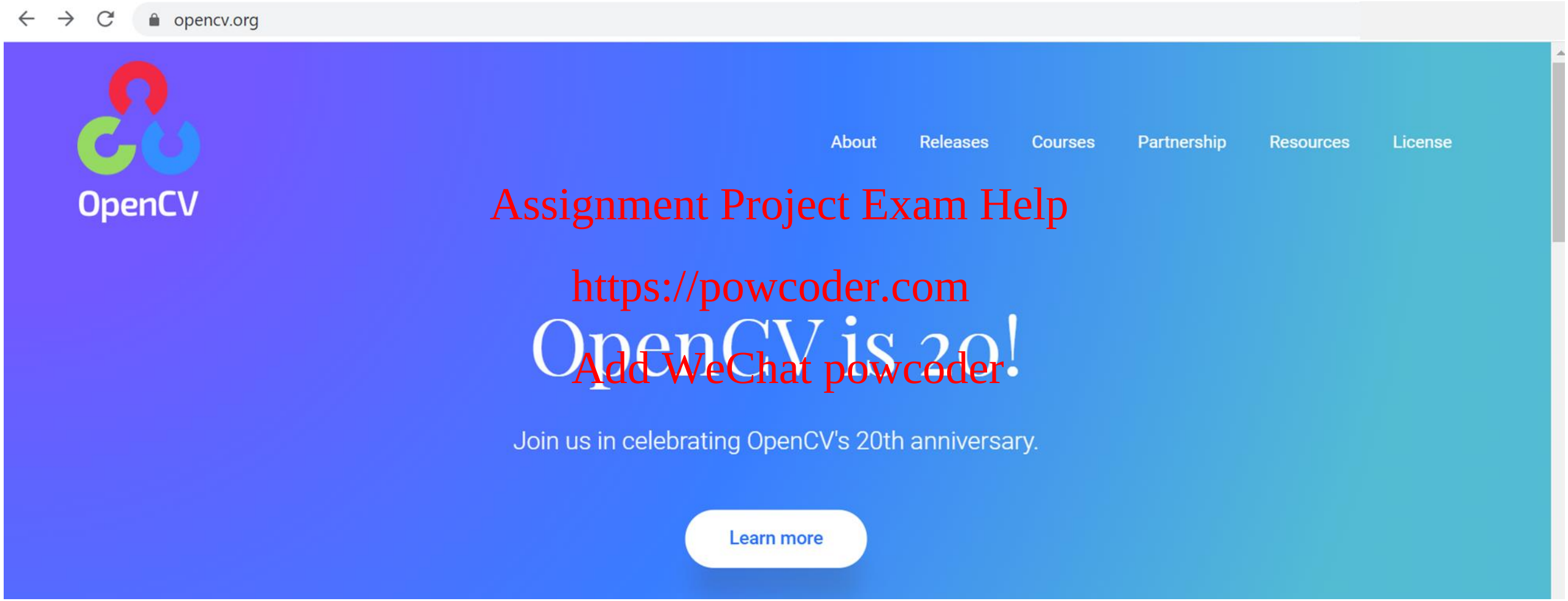
WEEK 1 - FURTHER READINGS

- [**2012_CVPR**] Accidental pinhole and pinspeck cameras
- [**1998_ICCV**] Bilateral Filtering for Gray and Color Images
- [**2004_IJCV**] Distinctive Image Features from Scale-Invariant Keypoints
- OpenCV Python Tutorials
- Python (numpy) / Matlab cheatsheet

Contents

- Setting up your playground
- Python
- **OpenCV**

OpenCV


opencv.org
OpenCV
About
Releases
Courses
Partnership
Resources
License
Assignment Project Exam Help
https://powcoder.com
OpenCV is 20!
Add WeChat powcoder
Join us in celebrating OpenCV's 20th anniversary.
Learn more

OpenCV

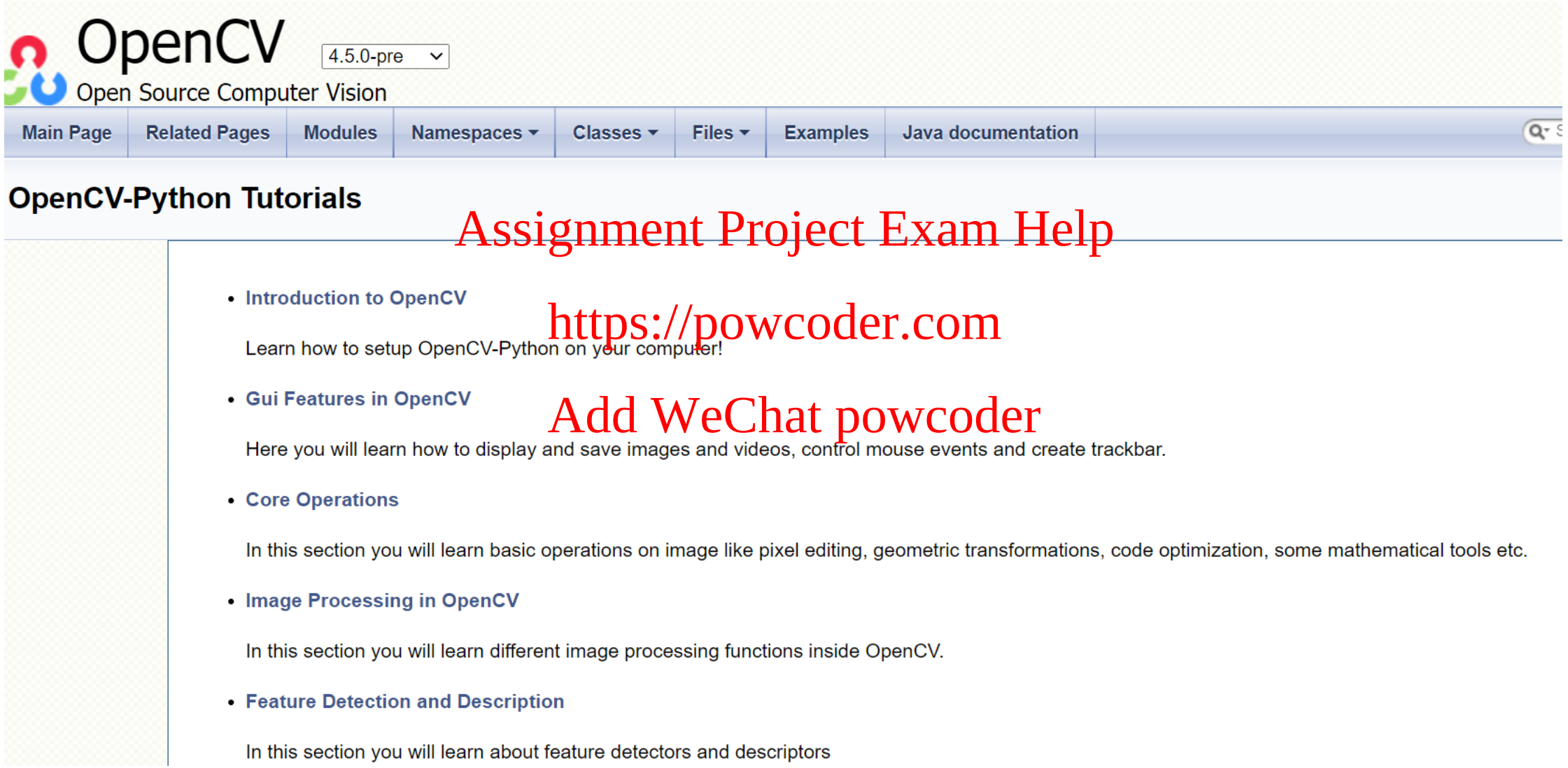


Image Read

- **cv2.imread()**
  - Loads an image from a file
  - If the image cannot be read (because of missing file, improper permissions, unsupport ed or invalid format), returns an empty matrix (Mat::data == NULL)

Python:

```
retval = cv.imread( filename[, flags] )
```

- **filename** – Name of file to be loaded
- **flags** – Flag that can take values of cv::ImreadModes


```
#include <opencv2/imgcodecs.hpp>
```

Loads an image from a file.

The function imread loads an image from the specified file and returns it. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format), the function returns an empty matrix (**Mat::data**==NULL).

Image Show

- **cv2.imshow()**
  - Displays an image in the specified window

◆ imshow()

```
void cv::imshow ( const String &  winname,
                  InputArray      mat
                )
```

Python:

```
None = cv.imshow( winname, mat )
```


Note

This function should be followed by **cv::waitKey** function which displays the image for specified milliseconds. Otherwise, it won't display the image. For example, **waitKey(0)** will display the window infinitely until any keypress (it is suitable for image display). **waitKey(25)** will display a frame for 25 ms, after which display will be automatically closed. (If you put it in a loop to read videos, it will display the video frame-by-frame)

Image Write

- **cv2.imwrite()**
  - Saves an image to a specified file

◆ imwrite()

```
bool cv::imwrite ( const String &          filename,
                   InputArray              img,
                   const std::vector< int > & params = std::vector< int >()
                 )
```

Python:

```
retval = cv.imwrite( filename, img[, params] )
```

More Info

WEEK 1 - FURTHER READINGS

- [**2012_CVPR**] Accidental pinhole and pinspeck cameras
- [**1998_ICCV**] Bilateral Filtering for Gray and Color Images
- [**2004_IJCV**] Distinctive Image Features from Scale-Invariant Keypoints
- OpenCV Python Tutorials
- Python (numpy) / Matlab cheatsheet

Example 1 – Image Read / Write

- Read Lena.jpg with colour and greyscale and show them

Example 1 – Image Read / Write

- **Read Lena.jpg with colour and greyscale and show them**

```
## image read, show, write
import cv2

im = cv2.imread('./inputs/lena.jpg')
im_gray1 = cv2.imread('./inputs/lena.jpg', 0)
im_gray2 = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

cv2.imshow('lena', im)
cv2.waitKey()

cv2.imshow('lena_gray', im_gray1)
cv2.waitKey()

cv2.imshow('lena_gray', im_gray2)
cv2.waitKey()

cv2.imwrite('./inputs/lena_gray.jpg', im_gray1)
```

Example 2 – Image Composition with Numpy

- Resize the Lena image to half resolution and composite as shown below:

Example 2 – Image Composition with Numpy

- Resize the Lena image to half resolution and composite as shown below:

```
# Image composition with numpy
import numpy as np
import cv2

im_gray = cv2.imread('./inputs/lena.jpg', 0)
h, w = im_gray.shape

h_resize = np.int(np.floor(h/2))
w_resize = np.int(np.floor(w/2))

im_gray_resized = cv2.resize(im_gray, (h_resize, w_resize))
im_composited = np.zeros([h, w])
im_composited[0:h_resize, 0:w_resize] = im_gray_resized

cv2.imshow('lena', im_gray)
cv2.waitKey()

cv2.imshow('lena_gray_com', np.uint8(im_composited))
cv2.waitKey()

cv2.imwrite('./inputs/lena_gray_small.jpg', np.uint8(im_composited))
```

Example 3 – Perspective Projection

- Given two point sets:
  - $\boldsymbol{x} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_4\} = \{(u_1, v_1), \dots, (u_4, v_4)\} = \{(41,176), (67,1133), (749,16), (749,1270)\}$
  - $\boldsymbol{x}' = \{\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_4\} = \{(u'_1, v'_1), \dots, (u'_4, v'_4)\} = \{(0,0), (0,1280), (749,0), (749,1280)\}$

Find the perspective projection matrix $\boldsymbol{P}$ such that $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$ and warp the image

?

Example 3 – Perspective Projection

```
rect = np.array([(176, 41), (1133, 67), (16, 749), (1270, 749)], dtype="float32")
dst = np.array([(0, 0), (1280, 0), (0, 749), (1280, 749)], dtype="float32")

M = cv2.getPerspectiveTransform(rect, dst)
warped = cv2.warpPerspective(im, M, (im.shape[1], im.shape[0]))
```


C: CvMat* **cvGetPerspectiveTransform**(const CvPoint2D32f* src, const CvPoint2D32f* **dst**, CvMat* **map_matrix**)

Python: cv.**GetPerspectiveTransform**(src, dst, mapMatrix) → None

Parameters:
- **src** – Coordinates of quadrangle vertices in the source image.
- **dst** – Coordinates of the corresponding quadrangle vertices in the destination image.

The function calculates the 3×3 matrix of a perspective transform so that:

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

where

$$\texttt{dst}(i) = (x'_i, y'_i), \texttt{src}(i) = (x_i, y_i), i = 0, 1, 2, 3$$

Example 3 – Perspective Projection

```
rect = np.array([(176, 41), (1133, 67), (16, 749), (1270, 749)], dtype="float32")
dst = np.array([(0, 0), (1280, 0), (0, 749), (1280, 749)], dtype="float32")

M = cv2.getPerspectiveTransform(rect, dst)
warped = cv2.warpPerspective(im, M, (im.shape[1], im.shape[0]))
```


Python: `cv.WarpPerspective`(src, dst, mapMatrix, flags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS, fillval=(0, 0, 0, 0)) → None

Parameters:

- **src** – input image.
- **dst** – output image that has the size `dsize` and the same type as `src` .
- **M** – 3×3 transformation matrix.
- **dsize** – size of the output image.
- **flags** – combination of interpolation methods (`INTER_LINEAR` or `INTER_NEAREST`) and the optional flag `WARP_INVERSE_MAP`, that sets `M` as the inverse transformation (`dst` → `src`).
- **borderMode** – pixel extrapolation method (`BORDER_CONSTANT` or `BORDER_REPLICATE`).
- **borderValue** – value used in case of a constant border; by default, it equals 0.

The function `warpPerspective` transforms the source image using the specified matrix:

$$\texttt{dst}(x,y) = \texttt{src}\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right)$$

when the flag `WARP_INVERSE_MAP` is set. Otherwise, the transformation is first inverted with `invert()` and then put in the formula above instead of `M` . The function cannot operate in-place.

Example 3 – Perspective Projection

- Given two point sets:
  - $\boldsymbol{x} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_4\} = \{(u_1, v_1), \dots, (u_4, v_4)\} = \{(41,176), (67,1133), (749,16), (749,1270)\}$
  - $\boldsymbol{x}' = \{\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_4\} = \{(u'_1, v'_1), \dots, (u'_4, v'_4)\} = \{(0,0), (0,1280), (749,0), (749,1280)\}$

Find the perspective projection matrix $\boldsymbol{P}$ such that $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$ and warp the image