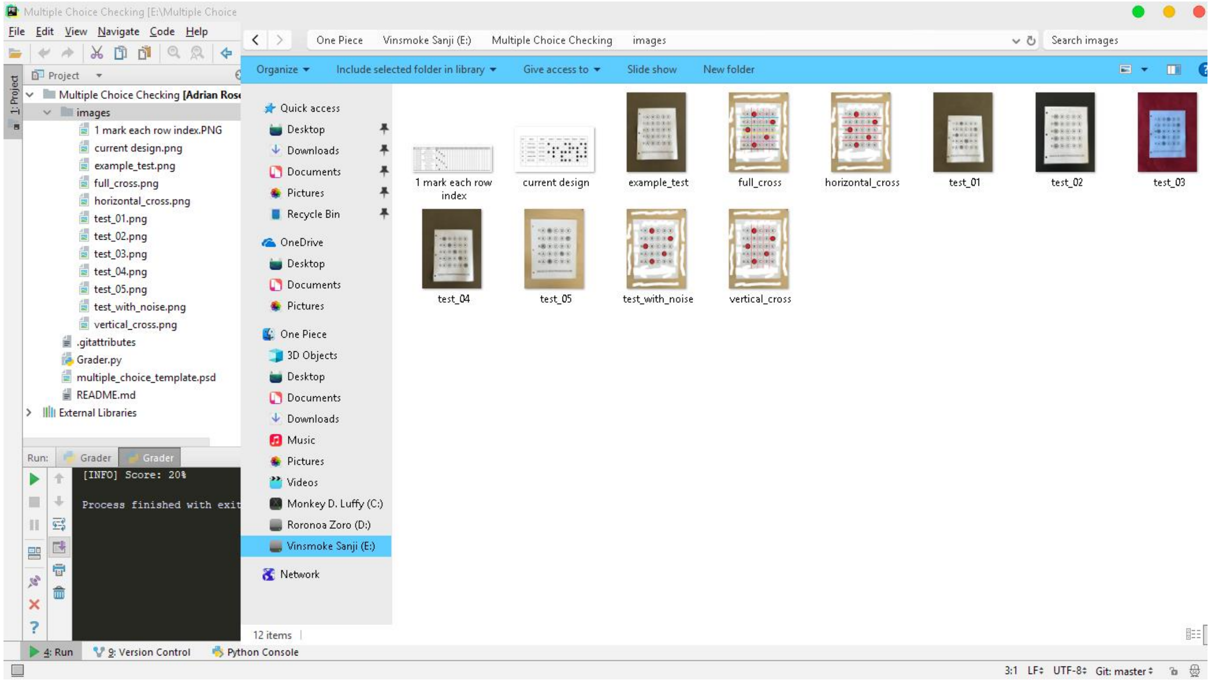
Multiple Choice Checking

Source: https://github.com/SynneKDoG/Multiple-Choice-Checking

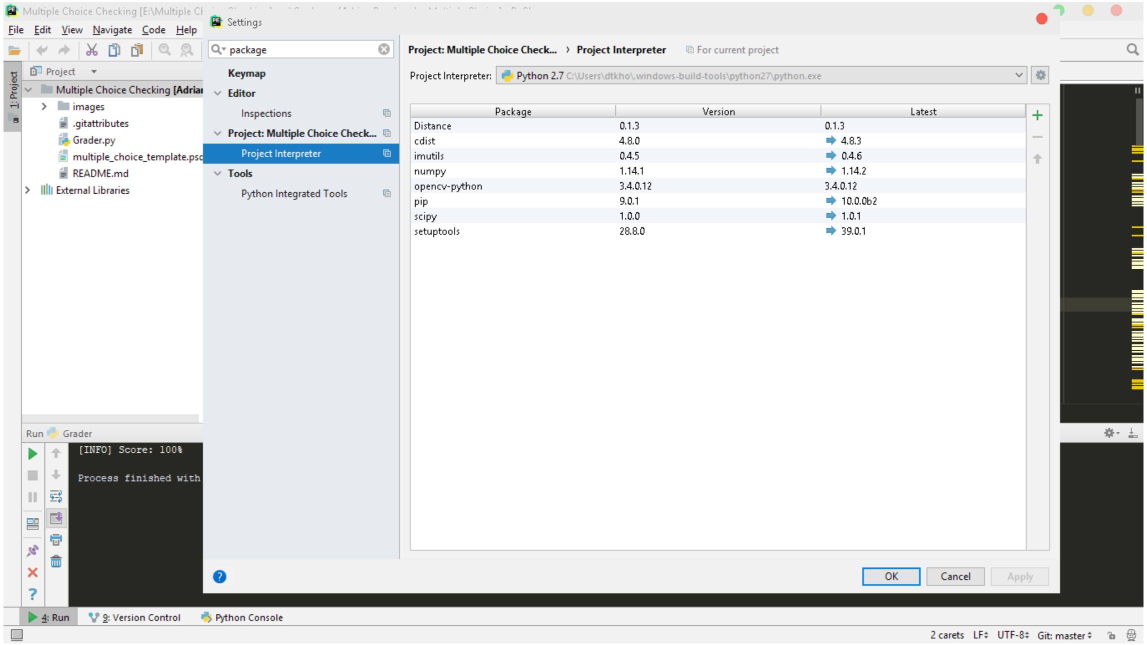
Tutorial: https://www.pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/

1) Your test images are here:



```
[INFO] Score: 20%

Process finished with exit
```

2) Install these package after you have set up OpenCV, Python and PyCharm successfully:


Multiple Choice Checking [E:\Multiple Ch
File Edit View Navigate Code Help
Project
Multiple Choice Checking [Adrian
images
.gitattributes
Grader.py
multiple_choice_template.psd
README.md
External Libraries
Settings
Keymap
Editor
Inspections
Project: Multiple Choice Check...
Project Interpreter
Tools
Python Integrated Tools
Project: Multiple Choice Check... › Project Interpreter
For current project
OK
Cancel
Apply
Run Grader
4: Run
9: Version Control
Python Console
2 carets LF÷ UTF-8÷ Git: master÷

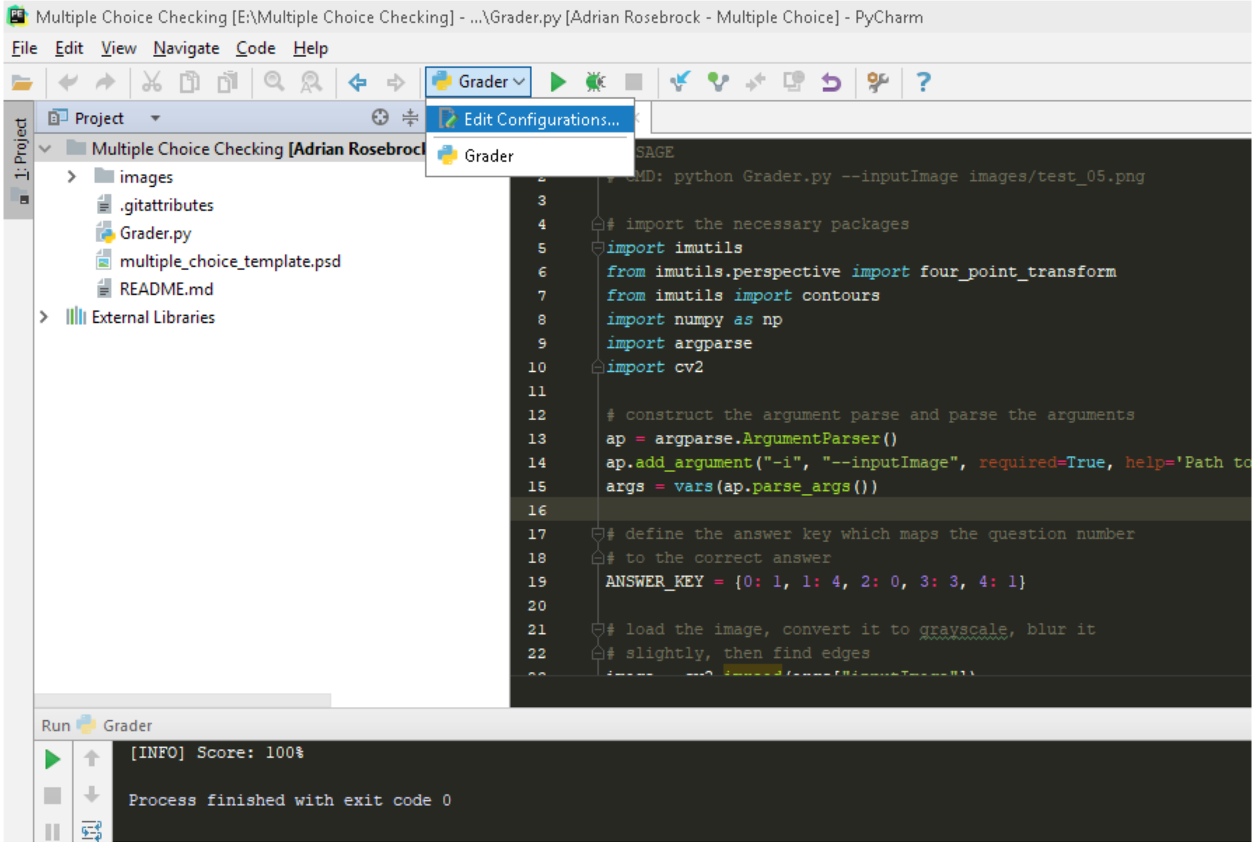
Settings search: package

Project Interpreter: Python 2.7 C:\Users\dtkho\.windows-build-tools\python27\python.exe

Package	Version	Latest
Distance	0.1.3	0.1.3
cdist	4.8.0	➡ 4.8.3
imutils	0.4.5	➡ 0.4.6
numpy	1.14.1	➡ 1.14.2
opencv-python	3.4.0.12	3.4.0.12
pip	9.0.1	➡ 10.0.0b2
scipy	1.0.0	➡ 1.0.1
setuptools	28.8.0	➡ 39.0.1

```
[INFO] Score: 100%

Process finished with
```

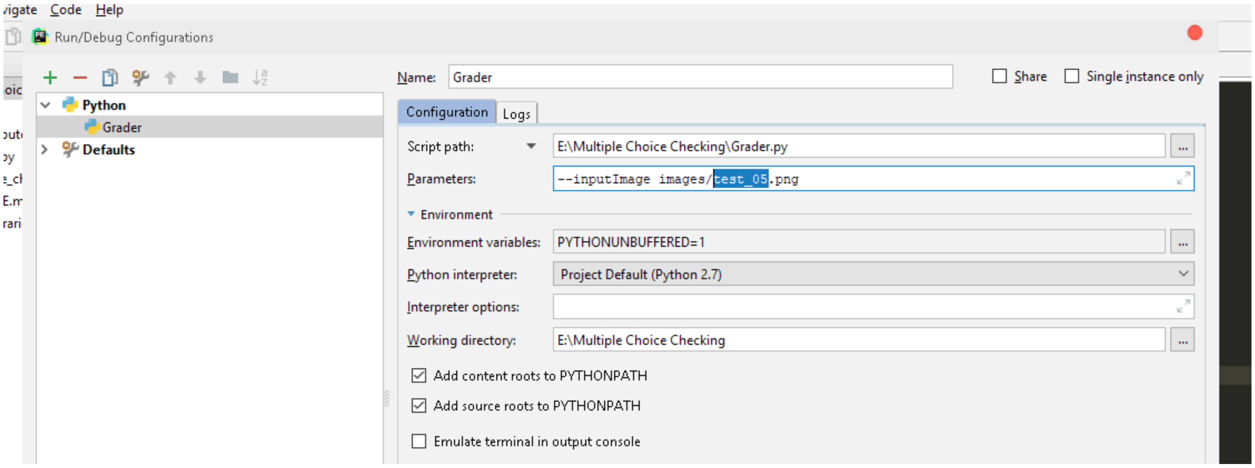
3) Edit Configurations to specify the image used for testing, it is referenced by ArgumentParser method:


Multiple Choice Checking [E:\Multiple Choice Checking] - ...\Grader.py [Adrian Rosebrock - Multiple Choice] - PyCharm
File Edit View Navigate Code Help
Grader
Edit Configurations...
Grader
1: Project
Project
Multiple Choice Checking [Adrian Rosebroc
images
.gitattributes
Grader.py
multiple_choice_template.psd
README.md
External Libraries
Run Grader

```
SAGE
# CMD: python Grader.py --inputImage images/test_05.png

# import the necessary packages
import imutils
from imutils.perspective import four_point_transform
from imutils import contours
import numpy as np
import argparse
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--inputImage", required=True, help='Path to
args = vars(ap.parse_args())

# define the answer key which maps the question number
# to the correct answer
ANSWER_KEY = {0: 1, 1: 4, 2: 0, 3: 3, 4: 1}

# load the image, convert it to grayscale, blur it
# slightly, then find edges
```

```
[INFO] Score: 100%

Process finished with exit code 0
```

4) Specify any name among those from step 1 (each one each case):



Name: Grader

□ Share □ Single instance only

Script path: E:\Multiple Choice Checking\Grader.py

Parameters: --inputImage images/test_05.png

Environment variables: PYTHONUNBUFFERED=1

Python interpreter: Project Default (Python 2.7)

Interpreter options:

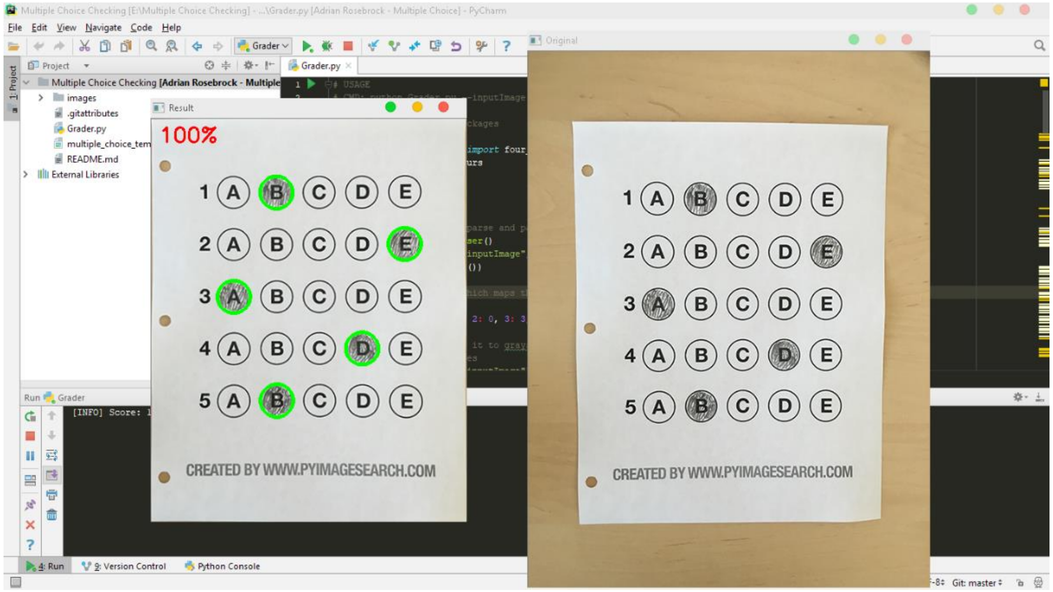
Working directory: E:\Multiple Choice Checking

☑ Add content roots to PYTHONPATH

☑ Add source roots to PYTHONPATH

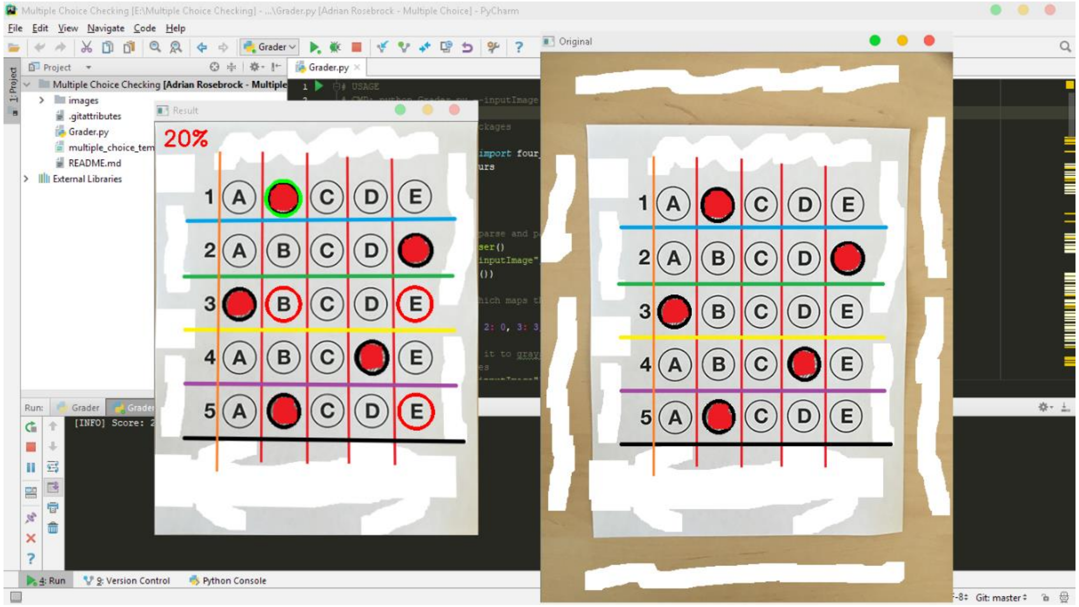
□ Emulate terminal in output console

5) Result:


Multiple Choice Checking [E:\Multiple Choice Checking] - ...\Grader.py [Adrian Rosebrock - Multiple Choice] - PyCharm
File Edit View Navigate Code Help
Grader
Project
Multiple Choice Checking [Adrian Rosebrock - Multiple
images
.gitattributes
Grader.py
multiple_choice_tem
README.md
External Libraries
Grader.py
Result
100%
1 A B C D E
2 A B C D E
3 A B C D E
4 A B C D E
5 A B C D E
CREATED BY WWW.PYIMAGESEARCH.COM
Original
Run Grader
4: Run
9: Version Control
Python Console
Git: master

```
[INFO] Score: 1
```

6) Other result for image on different case:



```
[INFO] Score: 2
```

Special case on “User marks multiple bubbles as correct for a single question”

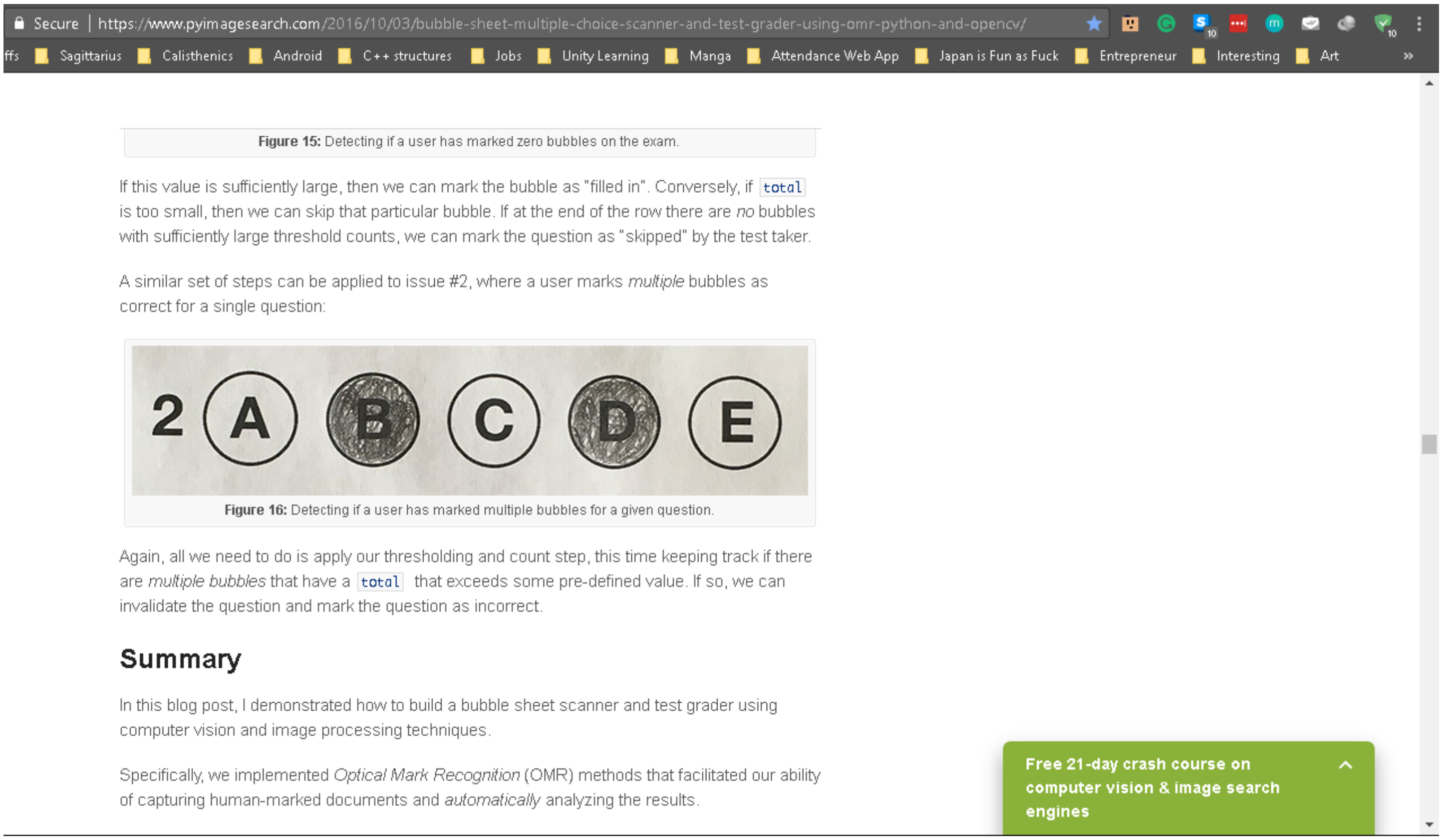

Secure | https://www.pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/
ffs
Sagittarius
Calisthenics
Android
C++ structures
Jobs
Unity Learning
Manga
Attendance Web App
Japan is Fun as Fuck
Entrepreneur
Interesting
Art
2 A B C D E
Free 21-day crash course on computer vision & image search engines

Figure 15: Detecting if a user has marked zero bubbles on the exam.

If this value is sufficiently large, then we can mark the bubble as "filled in". Conversely, if `total` is too small, then we can skip that particular bubble. If at the end of the row there are *no* bubbles with sufficiently large threshold counts, we can mark the question as "skipped" by the test taker.

A similar set of steps can be applied to issue #2, where a user marks *multiple* bubbles as correct for a single question:

Figure 16: Detecting if a user has marked multiple bubbles for a given question.

Again, all we need to do is apply our thresholding and count step, this time keeping track if there are *multiple bubbles* that have a `total` that exceeds some pre-defined value. If so, we can invalidate the question and mark the question as incorrect.

Summary

In this blog post, I demonstrated how to build a bubble sheet scanner and test grader using computer vision and image processing techniques.

Specifically, we implemented *Optical Mark Recognition* (OMR) methods that facilitated our ability of capturing human-marked documents and *automatically* analyzing the results.

If you have any problem, please contact: https://www.facebook.com/can.cao.tri

Just so you know: Like hell we would answer ridiculous questions -_________-