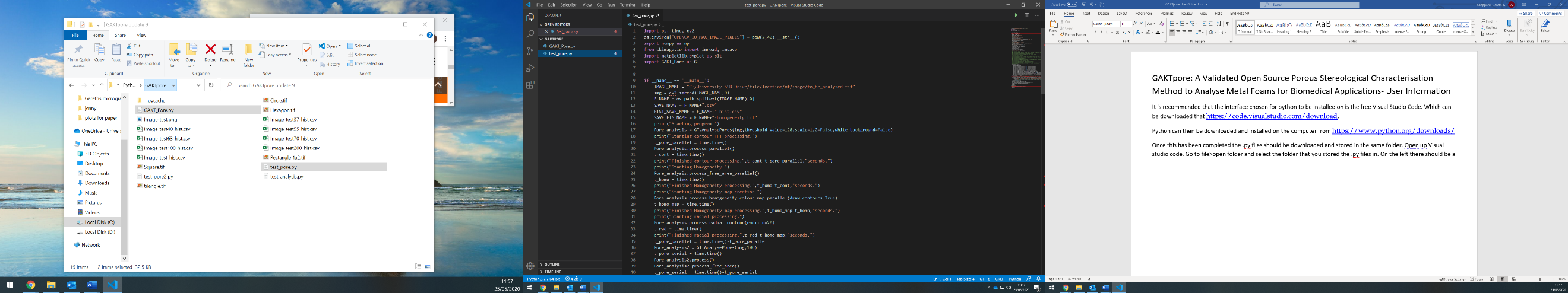
GAKTpore: Stereological Characterisation Methods for Porous Metal Foams in Biomedical Applications- User Information

It is recommended that the interface chosen for python to be installed on is the free Visual Studio Code. Which can be downloaded that <https://code.visualstudio.com/download>.

Python can then be downloaded and installed on the computer from <https://www.python.org/downloads/>

Once this has been completed the .py files should be downloaded and stored in the same folder. Open up Visual studio code. Go to file>open folder and select the folder that you stored the .py files in. On the left there should be an explorer which allows you to toggle between the two files.

All of the packages that are used by the algorithm now need to be downloaded. These are the packages that are at the top of **both** codes and their name is displayed after import.

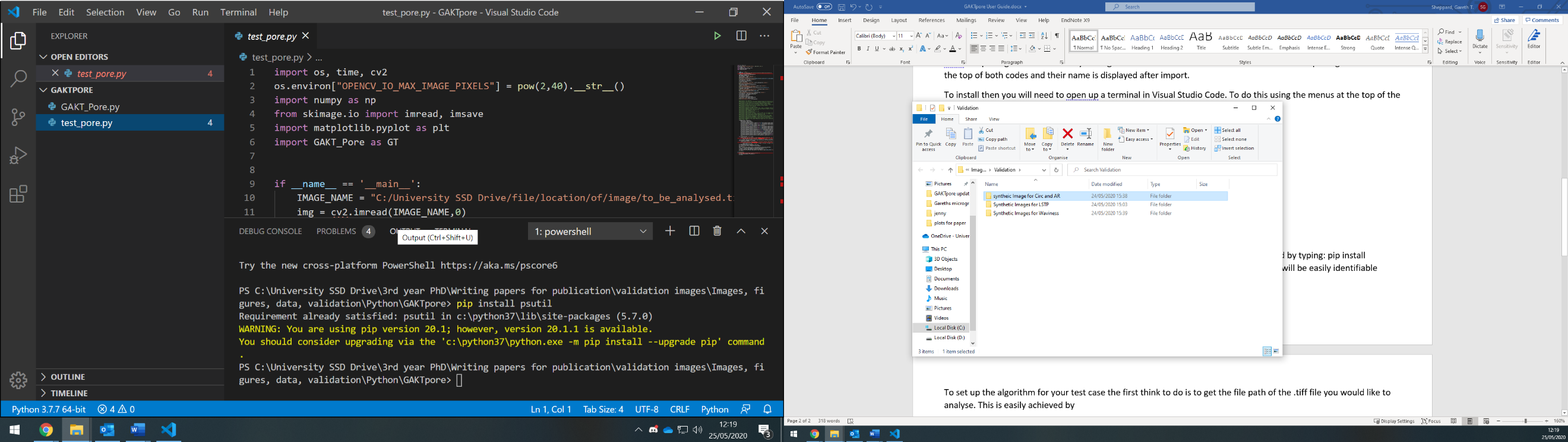
To install then you will need to open up a terminal in Visual Studio Code. To do this using the menus at the top of the screen. Terminal> New Terminal.

Then type into the terminal. –

* pip install --upgrade sciki
* pip install --upgrade python-opencv2
* pip install --upgrade opencv-python
* pip install --upgrade multiprocessing
* pip install --upgrade numpy, scipy, matplotlib
* pip install --upgrade wheel
* pip install --upgrade skimage
* pip install –upgrade psutil

If you still have a red line under any of the imported packages then this can be resolved by typing: pip install “package name”. If this does not work, then copy and paste the package name into google. There will be easily identifiable documentation displaying the command to use to download in the terminal Visual Studio Code.

The algorithm is now ready to use. The test\_pore.py is the file used to setting up the analysis and the running of the analysis.

To set up the algorithm for your test case(in **test\_pore.py**), the first think to do is to get the file path of the .tiff file you would like to analyse. This is easily achieved by opening the folder where the .tiff file is stored and clicking on it. There is then an option in the menu to copy the file path.

The file path can then be pasted in between the speech marks after IMAGE\_NAME = (**line 10**). Note that the slashes have changed from a forward slash(/) to a backwards slash(\). They will need to be changed back to a forward slash for the program to find the file. The algorithm will save all outputted data to the same folder where the file analysed is.

Next navigate down to **line 17** where you will find;

Pore\_analysis = GT.AnalysePores(img,threshold\_value=128,scale=1,G=False,white\_background=False)

There are four options that need to be entered/selected here. First is the image threshold- This is the thresholded at which the image will be binarised. This can be determined in ImageJ. Drag image from folder into ImageJ. (If imagej will not open the file then you will need to resave the image using the resave.py and use the new resaved image from then on.) Menu>Image>Adjust>Threshold. Move the bar around until a value is determined that covers the feature you want to analyse adequately. The scale can be determined by measuring the number of pixels across the scale bar. Select the line tool and draw a line across the length of the scale bar. Click on the analyze menu> measure. A box should pop out with the length measurement you need. Enter into the scale your scale bar/pixel length of scale bar. E.g 200 μm, 260 pixel. scale= 200/260. G is for a gaussian filter which can be used to help smooth very rough features in images. The white background features allows the algorithm knows what it is analysing. Depending on the colour of the background- true if the background image is white and false if its black.

Next navigate to **line 32**

Pore\_analysis.process\_radial\_contour(radii\_n=20)

The number 20 defines the number of ringed segments that the image will have. This is used to allow features of the image to be analysed in regards to distance. E.g a 10 mm sample with 20 ringed segments will have a step length between each segment of 250 μm.

Once this has all been done you can now run the algorithm by typing: python -i .\test-pore.py into the terminal.

Once the algorithm has finished it will be signified by the printout of the total processing time in seconds and then shortly after a homogeneity map will pop out for inspection. When ready, close the map and enter Ctrl Z to leave the data set. The algorithm is now ready to be run again.

Homogeneity map:

It is not recommended to change anything in the GAKTpore.py code, but if required the homogeneity colour map can have the scale bar range and colour scheme changed in there.

Scale bar range is **line 324**. Vmin= minimum value and Vmax= largest value- range is currently set 0 to 1.

Map colour scheme is **line 327**. Currently set as “jet”. More colour schemes can be found by searching for cmap colours. <https://matplotlib.org/3.1.1/tutorials/colors/colormaps.html>

Standard colour schemes are hsv, twilight, rainbow and terrain.

Problem shooting:

* do you have the right folder open in the terminal? need to have the folder open that the .py files are in.
* are all of the imported packages downloaded? Try redownloading
* is the IMAGE\_NAMEE file path correct? Bing = imag.copy() usually pops up as an indication of incorrect image path
* Is the image open with another application? Close it
* The accuracy of the algorithm is very good but the outputted data is only as good as the image segmentation. More pixels= more detailed image= easier to extract wanted features
* Problem- the homogeneity map displays a large free space that is not in the original image. Issue- very large contours are filtered out as they are usually errors. First the thresholding should be looked at to see if by optimising the value this can be avoided. If not the contour size filter can be changed to a larger value- **Line 68** in the GAKTpore.py code. It is set at 0.8- increasing this value will allow larger contours to be accepted.