SECTION 16: WORKING WITH IMAGES WITH PYTHON - 24 minutes, 4 parts

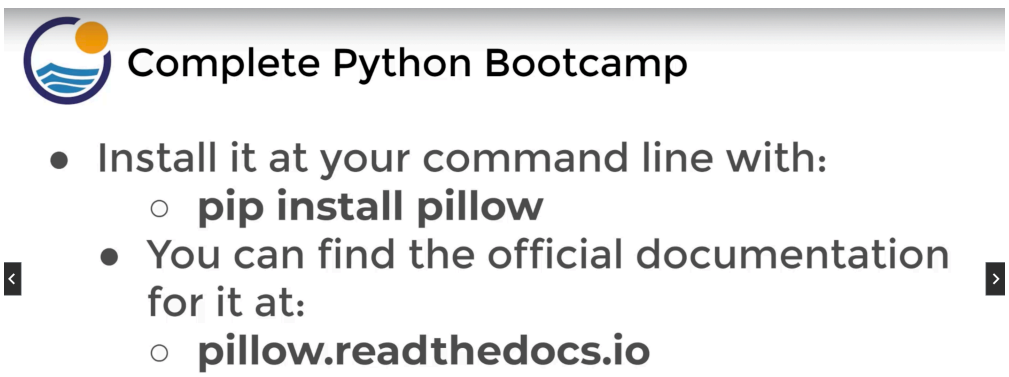
2/4 Working with Images with Python

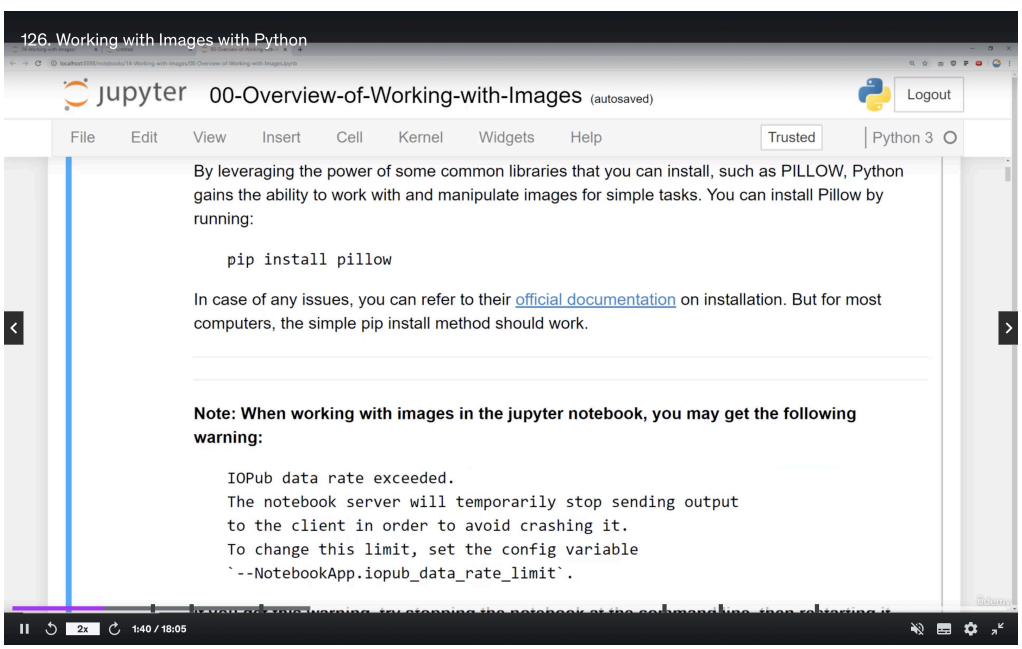
- **overview**
  - -> working with images in Python
  - -> the Pillow library <- the Python imaging library
  - -> this has function calls
  - **-> this is a library for image processing in Python which needs installing**

- **installation**
  - -> pip install pillow <- in the terminal
  - -> there is documentation for this

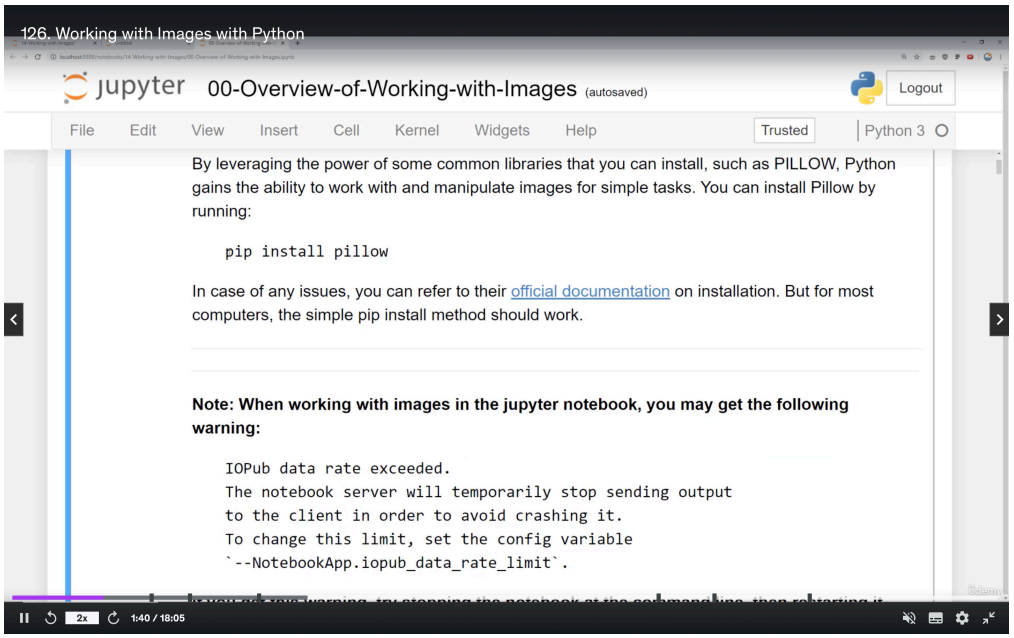
- -> there are three notebooks for this section
- -> there are multiple different image files which we are going to use in the ipynb file
- -> installing pillow

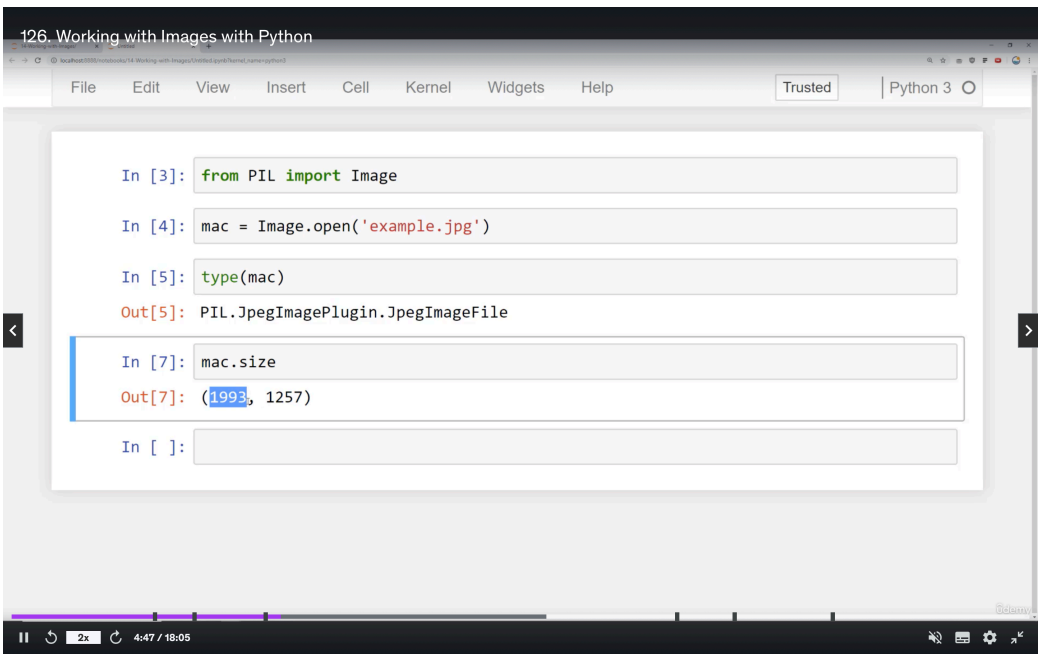
- **data rate limits**
  - -> the data rate can be exceeded
  - -> there is a default data rate limit
  - -> if this is exceeded, then we restart it
  - -> more information about this is on Stack Overflow

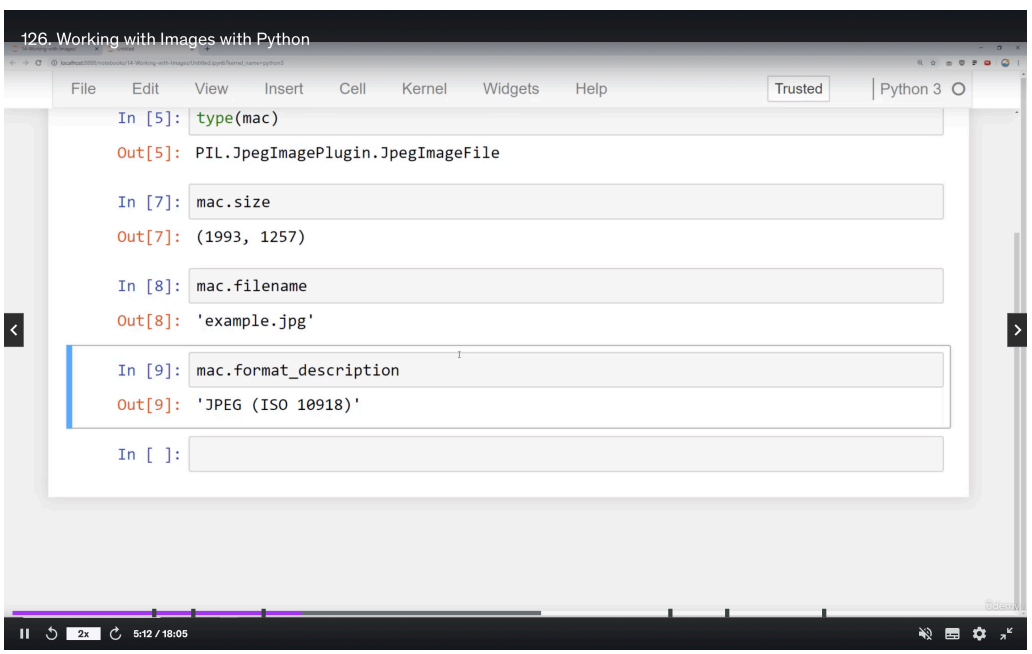


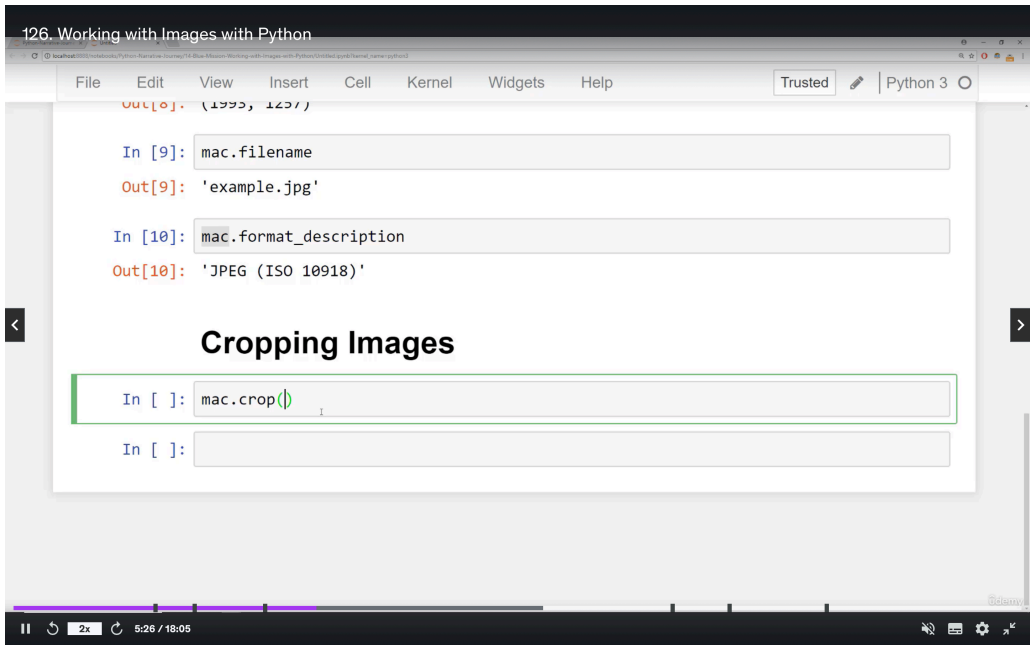


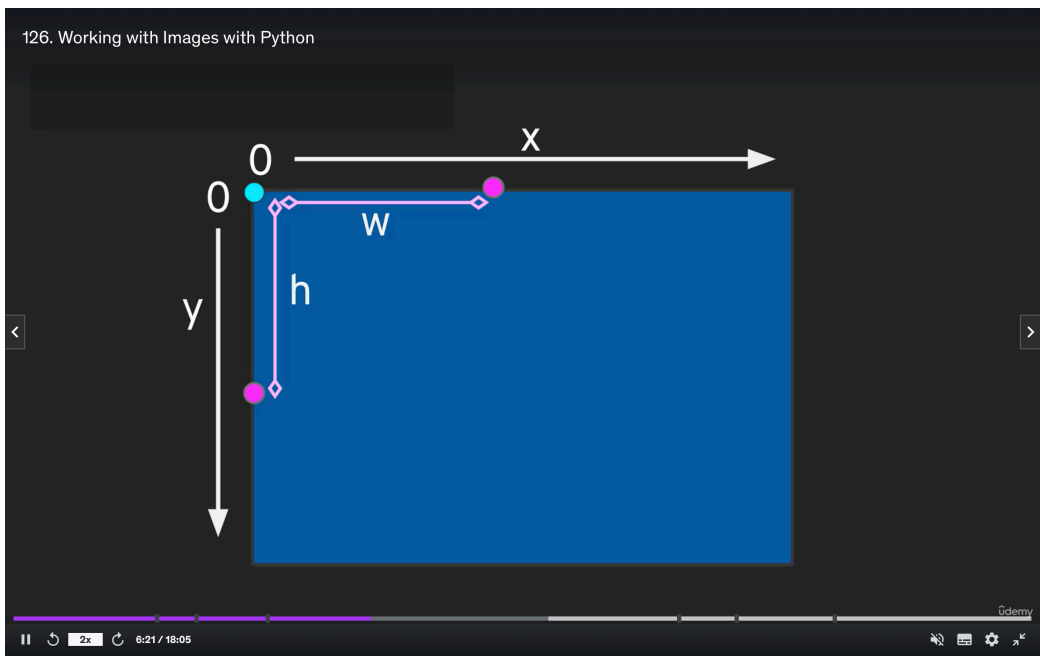
- **opening images with Python**
  - import modules
    - -> from PIL import Image <- importing the Image class from the PIL module / library
  - **open the image and store it in a variable**
    - -> he then defines another variable to open an image
    - -> he prints returns the type of file
    - -> this is a plugin
    - -> there are multiple different file types
  - **process the image**
    - -> he prints out one of the image files in the ipynb file
    - -> we have an ipynb file, and multiple image files
    - -> he has opened one of the image files in the ipynb file and stored it in a variable
    - **-> now he is printing out that variable <- returning the image**
      - -> this can be done by typing the name of the variable which stores the image
      - -> or by its_name.show()
    - **-> then printing out more information associated with it**
      - -> its size
      - -> the file name



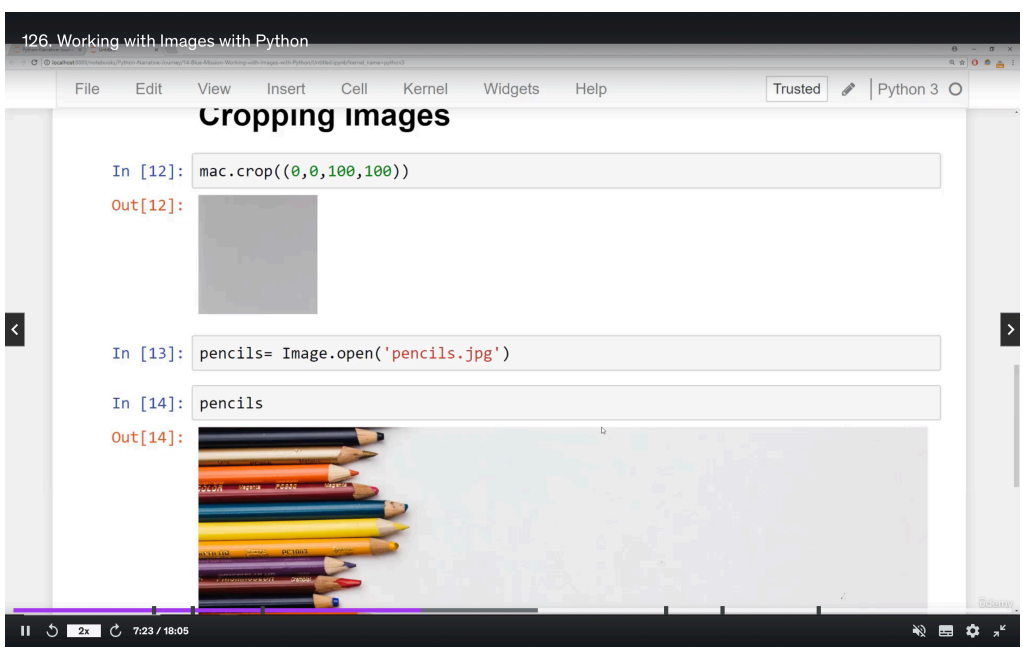




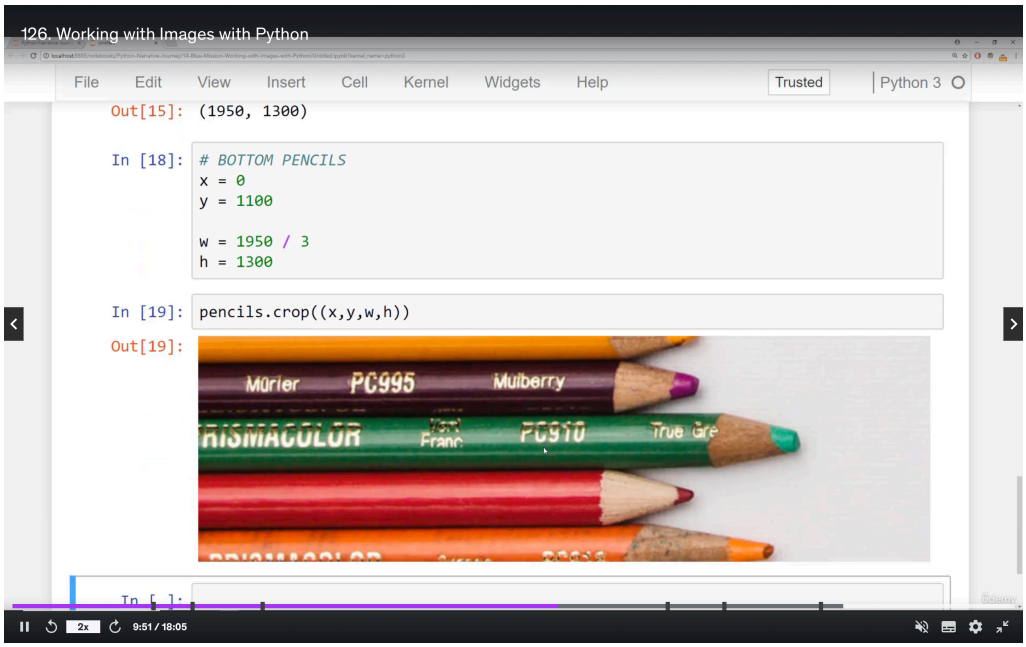


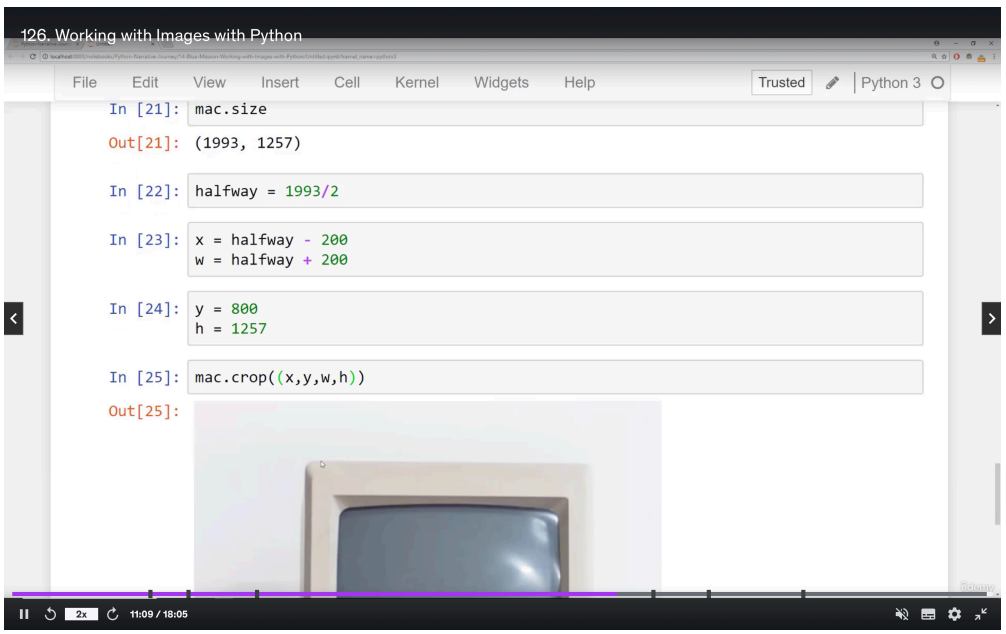


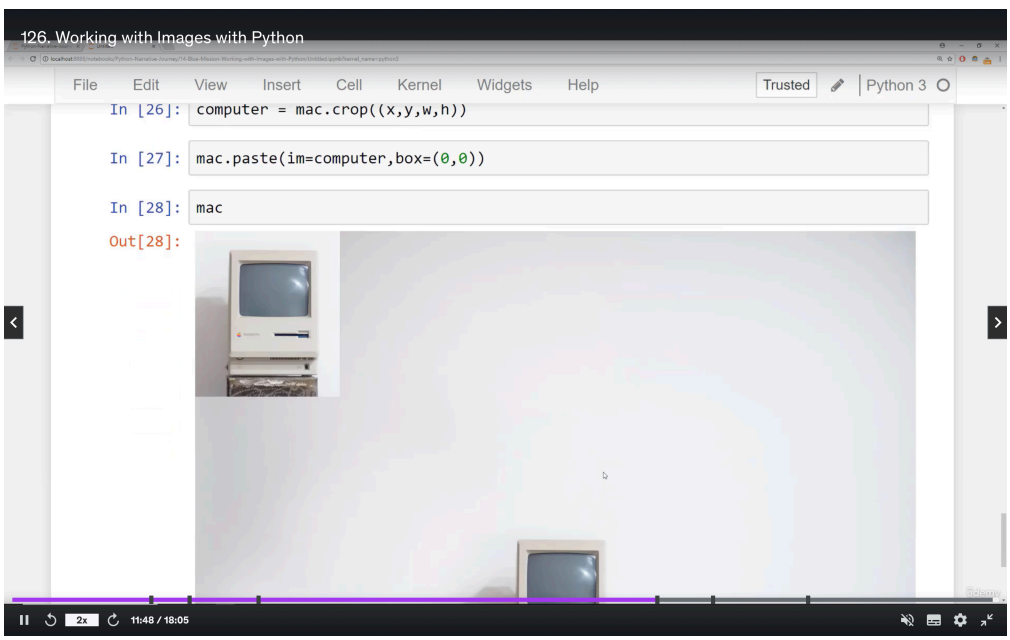
- **-> cropping images**
  - -> variable_name.crop()
  - -> this returns coordinates
  - -> this by the second quadrant on a x/y graph
  - -> on this coordinate, you can create a rectangle and crop out an image
  - -> this is done with variable_name.crop
  - -> the name of the variable which stores the image
  - -> pencil example

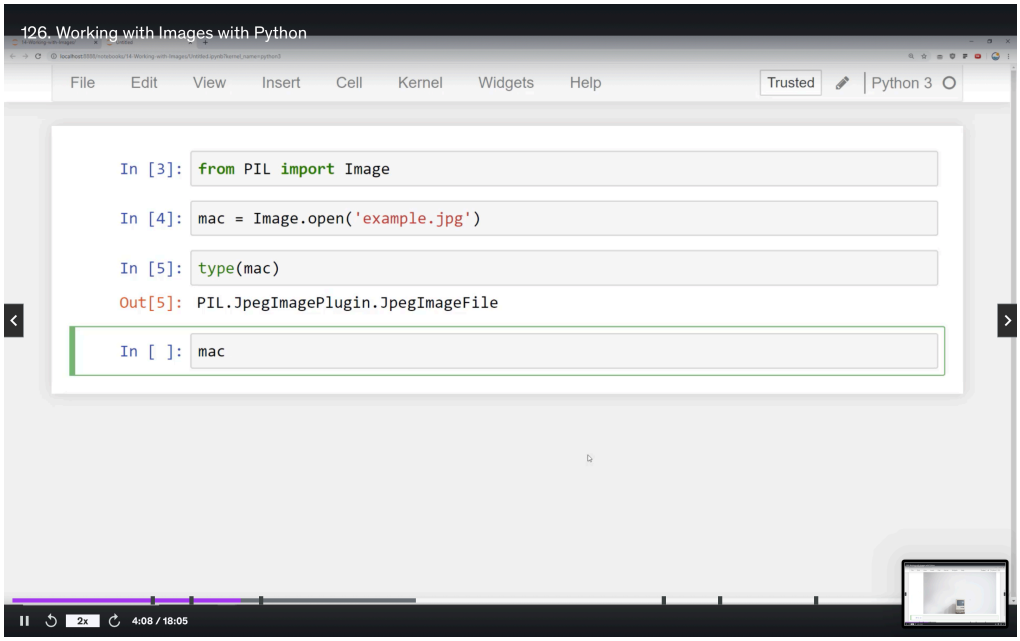


    - -> he has imported a pencil image and stored it in a variable
    - -> then returned the variable
    - -> he sets the x and y coordinates (stores values for these in two variables)
    - -> then sets the width and height which we want for the crop
- -> mac example
    - -> he prints out the imported image of a mac in the ipynb file
    - -> then the size of this, using .size
    - **-> then to crop it, he defines the aspect ratio which we want as a function of this height**
    - -> he sets the x and y coordinates which we want to crop
    - -> knowing these requires a trial and error

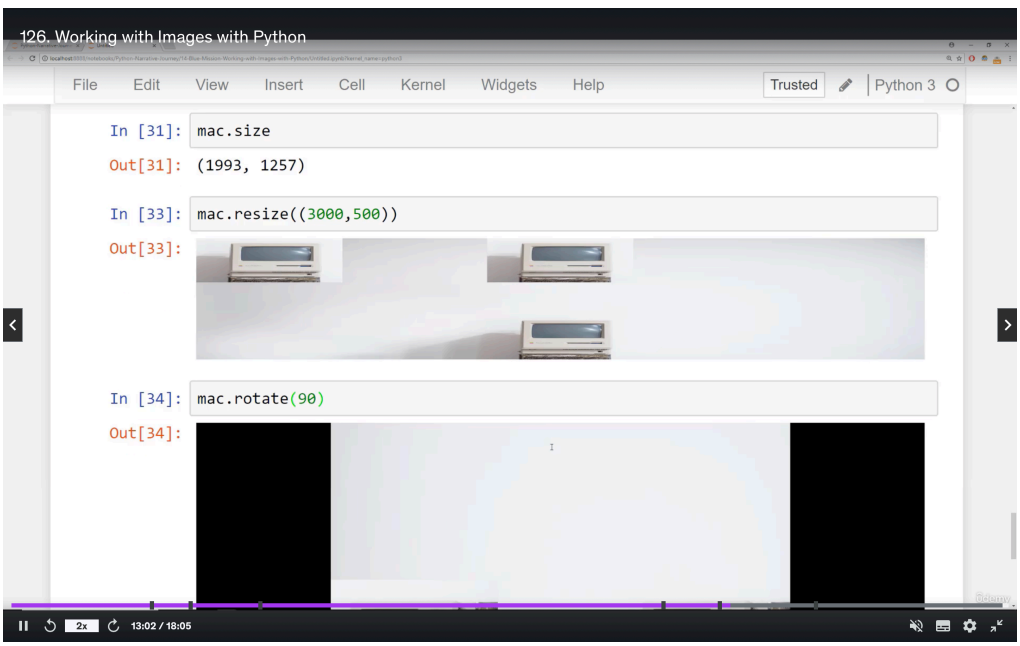


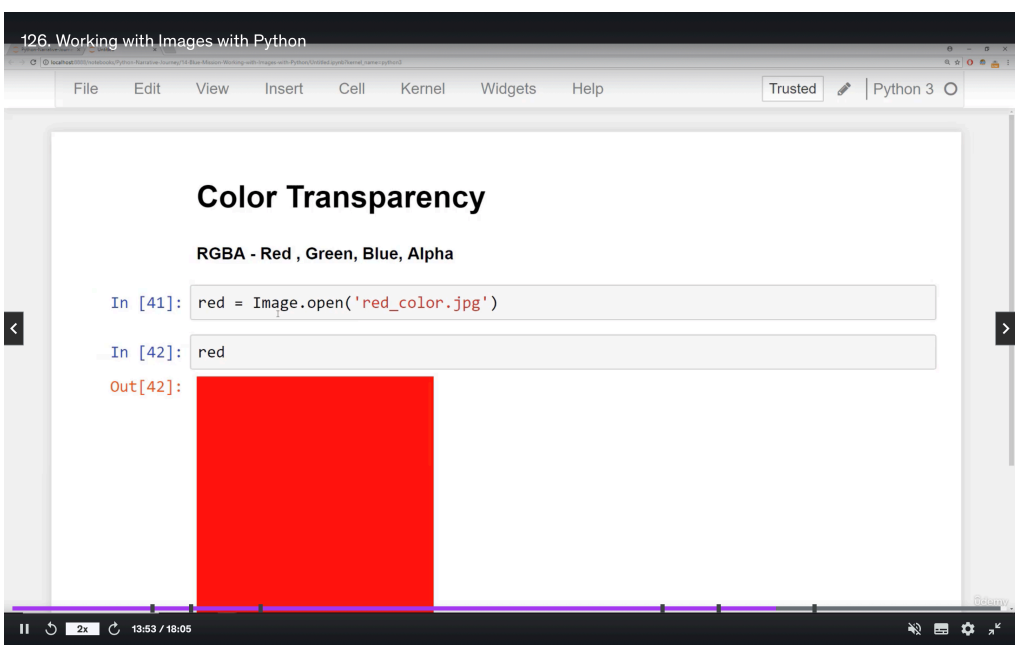






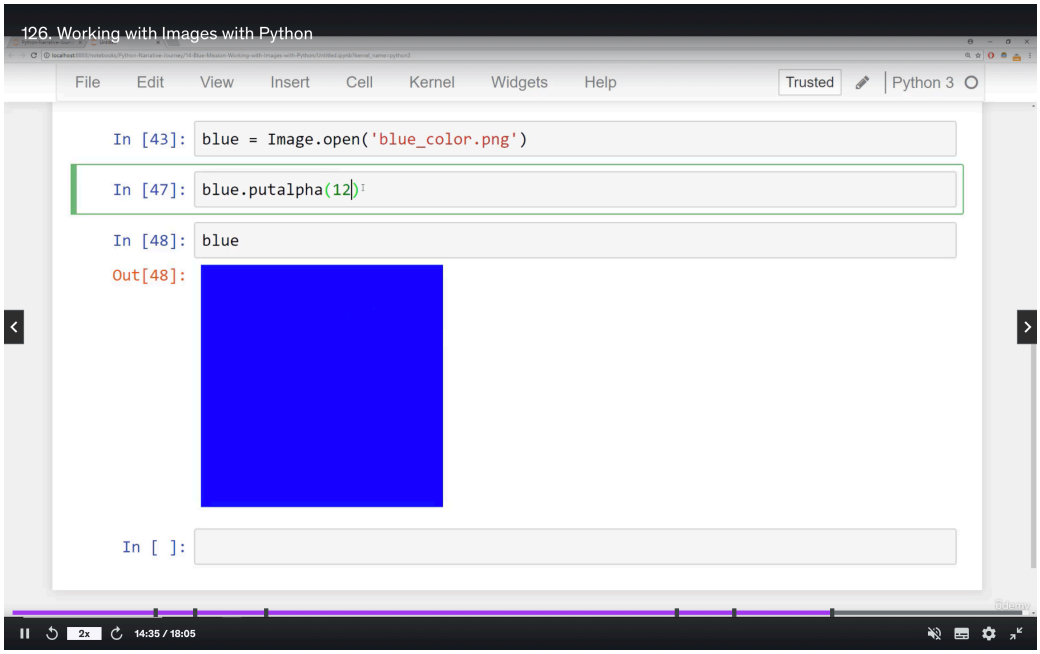
- **-> copying and pasting images**
  - -> he stores an image in a variable
  - -> then uses the .paste method with the image as its argument
  - -> he's copied an image and pasted it over the top of another one
  - -> he's set the coordinates on the second image where he wants the first one pasted
  - -> multiple images can be copy and pasted over the same one



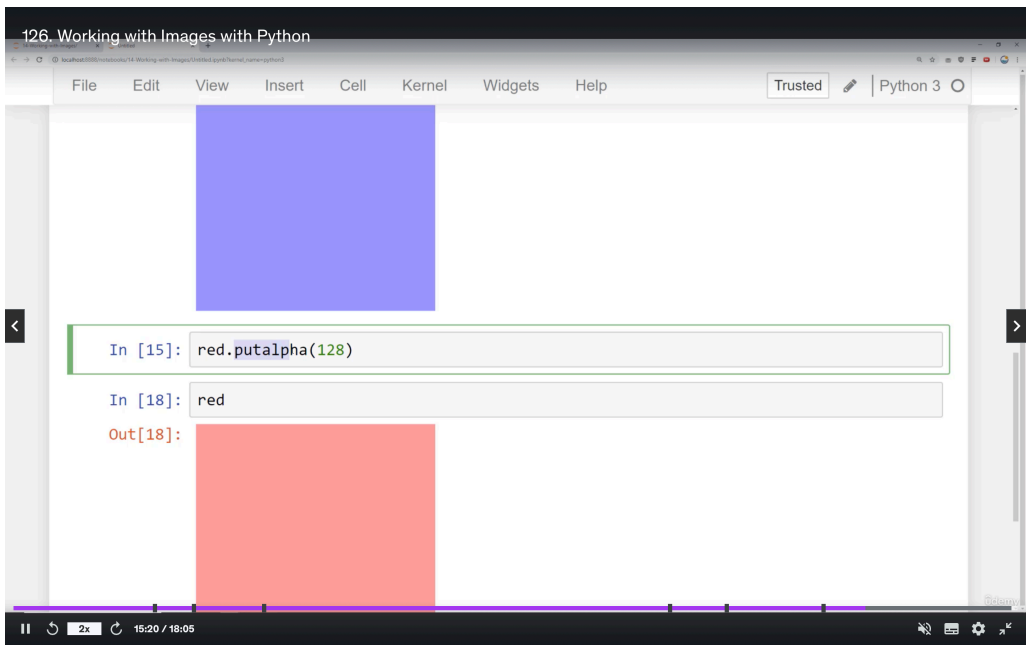


- **-> the resize method**
  - -> we can print the size of an image
  - -> the image is stored in the name of a variable
  - -> the .resize method is used on this, with the argument the dimensions of the new image which we want

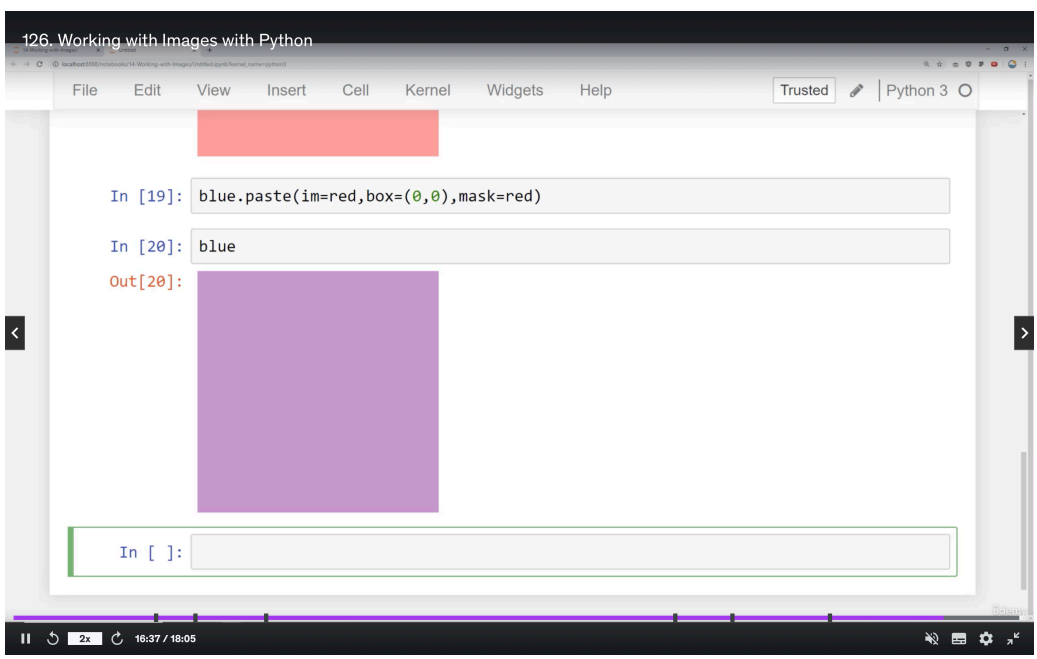
- -> you can also use variable_name.rotate



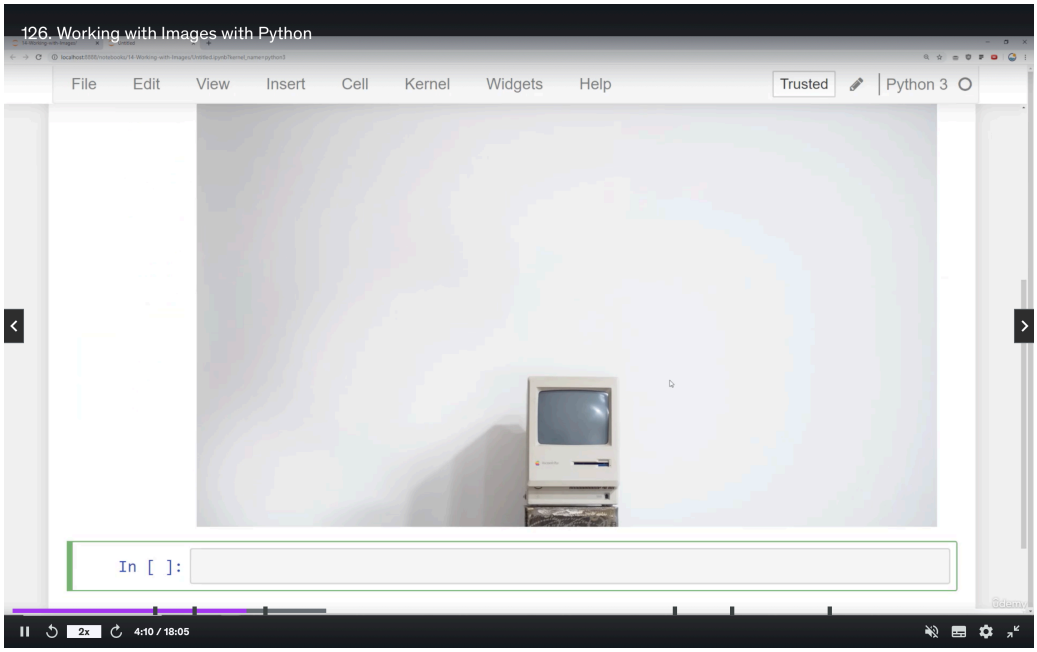
- **-> colour transparency**
  - -> images have rgba <- a is alpha
  - -> these go from 0-255
  - -> alpha is 0, then the image is transparent
  - -> 255 is opaque



  - **-> changing alpha**
    - -> we have an example of an image which is stored in a variable
    - -> using .putalpha(n) on this variable can be used to transform the transparency of the image
    - -> n is the value of alpha



- **-> mixing two colours together**
  - -> pasting one image on top of another
  - -> this is done with the .paste method
  - -> one of the images goes on top and the other on the bottom



  - -> the arguments are the current image object, the box (alignment) and the mask (this should be the same as the image)
  - -> the image we want on top, then how we want to align it based on the top left, then the mask being the same as the image
  - -> knowing the image and where you want to place them
  - -> to save the resulting image
    - -> the .save method
    - -> the argument of this should be the path where we want the image to be saved
    - -> the image can't already exist elsewhere