

# Unit: Images, memes and face filters

## Summary

You will use the Python programming language to explore how apps like Instagram and Snapchat create their various filters and effects on your photos.

This unit assumes some existing knowledge of Python basics.

Last updated: 02/01/2020

## Unit information



MYP item	This unit
Key concept	Communication
Related concepts	Adaptation, form
Global context	Personal and cultural expression
Statement of inquiry	Technology allows us to adapt our form when communicating with different audiences to more creatively express ourselves
Assessment objectives	A (inquiring & analysing), B (devising ideas), D (evaluating)

## Lesson overviews

12 lessons as follows:

1. Photo basics: Use camera to take photos, open photos from disk, save photos to disk
2. Photo manipulation: Crop, resize, rotate, photo in photo
3. Simple filters: Black and white, re-colour, blur
4. Create your own memes
5. Object detection: faces, bodies, facial features
6. Complete the face filter demo
7. Design your own face filter
8. Build your own face filter
9. Evaluate your own face filter

## Website

The website for this unit is <https://pbaumgarten.com/myp-design/image-editing/>

# 0. Prerequisites

---

## Python installed

This assumes you have a recent version of Python installed, typically at least version 3.6

If you don't have it, go to <https://www.python.org> and download it.

When running the installer, make sure you turn on the option to "Add Python to PATH"

I have a video walk through of the process for installing Python and VS Code at

- <https://pbaumgarten.com/python/install/>, or
- <https://youtu.be/-R6HFLp7tTs>

## Libraries required

Once you have Python installed, open the command prompt and run the following

```
pip install Pillow ImageToolsMadeEasy
```

If you get a permissions error with the above, try it again with the `--user` switch as follows

```
pip install --user Pillow ImageToolsMadeEasy
```

## Basic Python knowledge

This guide assumes a basic familiarity with Python. I have written a quick recap designed for a one hour lesson should you need it. It is available at:

- <https://pbaumgarten.com/python/recap/>

If you need a more detailed introduction to Python I have a set of detailed tutorials on my website. Each lesson contains detailed notes, videos and practice exercises. Each lesson is roughly an hour in length with 9 lessons in "the basics" (though only the first 5 are required for this tutorial).

- <https://pbaumgarten.com/python/>

# 1. Photo basics

---

The following section requires the following imports

```
from PIL import Image
import ImageTools
```

## Use camera to take a photo image

A video walkthrough of getting this working in VS Code is available here [https://www.youtube.com/watch?v=Lj\\_mHL3EA\\_Y](https://www.youtube.com/watch?v=Lj_mHL3EA_Y) or via the QR code.

To take a photo, you create an instance of the camera object, and then you can use the built in `.take_photo()` function to trigger the camera and return an Image object. It may take a second or two for this line to execute depending on the speed of your built in camera.

Once you have your Image object (called `img` in my example below), it has built in commands that you can use to save it to a PNG or JPG image file, and/or open a preview window to display it.



A basic example

```
camera = ImageTools.Camera()
img = camera.take_photo()
img.save("my photo.jpg", "jpeg")
img.show()
```

- Note the `save()` command requires two parameters. The first is the filename, the second is a predefined code indicating we want to save as a JPEG format. It must be spelt with the 'e' in 'jpeg'. You can also use 'png' and a bunch of others.

## Open an image

If you already have an existing PNG or JPG you wish to open...

- The image must be in the same folder as your Python project.

```
img = Image.open("my picture.jpg")
img.show()
```

## Save an image

As already shown above section, once you have an Image object it has a built in save command. It can save in PNG or JPG format. It will be saved into your project folder.

```
img = Image.open("my picture.png") # alternatively use the camera
img.save("myphoto new copy.png", "png")
img.save("myphoto another copy.jpg", "jpeg")
```

## Get image information

The Image object will advise you the width and height of your image in pixels, and can inform you of the colour mode for your image. The `img.size` attribute returns both the width and height so to access it use two variables like...

```
width, height = img.size
mode = img.mode
```

The mode refers to the colour scheme in use. We will use this mode information in later lessons to switch between different colour modes. The ones you are most likely to have use for are...

- **1** (1-bit pixels, black and white, stored with one pixel per byte)
- **L** (8-bit pixels, black and white)
- **P** (8-bit pixels, mapped to any other mode using a color palette)
- **RGB** (3x8-bit pixels, true color)
- **RGBA** (4x8-bit pixels, true color with transparency mask)
- **CMYK** (4x8-bit pixels, color separation)
- **HSV** (3x8-bit pixels, Hue, Saturation, Value color space)

The following is a handy one line print statement I use to print a summary of my Image information after taking a photo or opening a new image.

```
print(f"Image info: Size {img.size[0]} x {img.size[1]}, colour mode {mode}")
```

## Your task/s

- Successfully get the Python Image system working
- Successfully open an existing PNG or JPG photo on your computer and have Python open it for viewing
- Obtain the information for your existing photo. What width, height and mode does Python say it has?
- Successfully have the camera take a photo
- Obtain the information for your camera photo. What width, height and mode does Python say it has?
- Successfully save the photo from your camera. Upload it to your portfolio.

## 2. Simple photo manipulation

---

The following section requires the following imports

```
from PIL import Image
import ImageTools
```

### Crop

The image object has a built in crop command that requires a set of 4 values denoting the pixel boundaries of the rectangle to crop the image down to. The 4 values are left-edge, top-edge, right-edge, and bottom-edge.

You can either supply the boundaries as a separate variable, or embed them within the crop command as shown. Note the extra set of parenthesis if you use the embedding approach. In either method you can use actual integers, or variables with integer values.

In both cases the original Image, `img`, is untouched. The cropped version of the image has been put into the `cropped_img` object in these examples.

```
# Method 1
img = Image.open("my picture.png") # alternatively use the camera
boundaries = (200, 50, 400, 200)
cropped_img = img.crop(boundaries)

# Method 2
img = Image.open("my picture.png") # alternatively use the camera
cropped_img = img.crop((200, 50, 400, 200))
```

### Resize

The Image object also contains a `.resize()` command that requires a 2 value set with the new width and height in pixels that you want to use. Again, the original image is untouched. You can use variables containing integer values, or actual integer numbers.

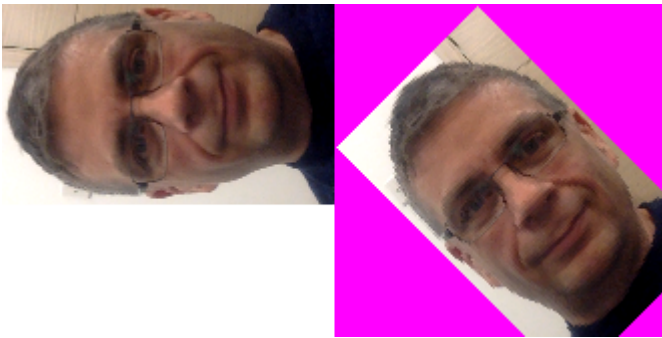
Example using variables

```
img = Image.open("my picture.png") # alternatively use the camera
resized_img = img.resize((new_width, new_height))
img.show()
```

Example using integer numbers

```
img = Image.open("my picture.png") # alternatively use the camera
resized_img = img.resize((300, 200))
img.show()
```

## Rotate



To rotate an image requires a number of degrees (anti-clockwise). While usually you may use 90, 180 or 270 you are not restricted to these.

By default the rotated image will be the same dimensions as the original. This means if you supply a portrait photo that is 800 x 600 pixels and rotate it 90 degrees, it will rotate and then appear cropped because the 800 pixel wide portion won't fit in the 600 pixel height you are putting it in. To overcome this, set `expand=True` and Python will enlarge (or shrink) the new image as required.

If there is any blank space in the new image (such as if you are rotating on 45 degrees and will end up with a triangle in each corner), you can specify the colour to use to fill the blank space with the `fillcolor` command.

```
# Example 1
rotated_img_1 = img.rotate(90, expand=True, fillcolor="#00ffff")
# Example 2
rotated_img_2 = img.rotate(45, expand=True, fillcolor="#ff00ff")
```

## Paste one image into another image

To paste one image into another image, simply use the `.paste` command as shown. You provide it the image to insert, and a set of (x,y) coordinates for where you'd like to overlay it into the original image.

```
img1 = Image.open("my picture.png") # alternatively use the camera
img2 = Image.open("my other picture.png")
img1.paste(img2, (100,100))
img1.show()
```

## Create a new, blank image

The `.new()` command needs two parameters: The first specifies the mode for this image, and the second is a set of (width,height) size dimensions.

Example

```
img = Image.new("RGBA", (3000, 2000))
```

You can create new blank images as the target to paste a bunch of other images into if you wish.

## Your task/s

- Take a photo of someone with your camera
- Take a crop of their face from that photo (approximately 100x180 pixels)
- Resize the original photo to the same size as the crop (approximately 320x180 pixels)
- Create a new blank image wide enough for both the crop and shrunk version of the original (approximately 420x180 pixels)
- Paste the crop photo and the resized/shrunk photo into the blank image
- Save and upload the result to your portfolio

An example of what you are aiming for...



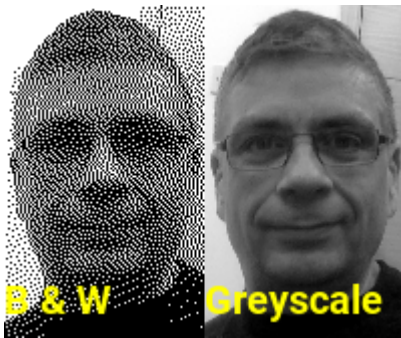
## 3. Simple filters

---

The following section requires the following imports

```
from PIL import Image, ImageFilter, ImageEnhance
from PIL import ImageFont
import ImageTools
```

### Convert to black and white or grey scale



The easiest way to convert to black and white or greyscale is to convert the image mode.

```
camera = ImageTools.Camera()
img = camera.take_photo()
bw = img.convert(mode="1")
bw.show()

# alternatively open an image file
# black and white
```

Greyscale works the same, we just set the mode to **L**.

```
img = Image.open("my picture.png")
grey = img.convert(mode="L")
grey.show()

# alternatively use the camera
# greyscale
```

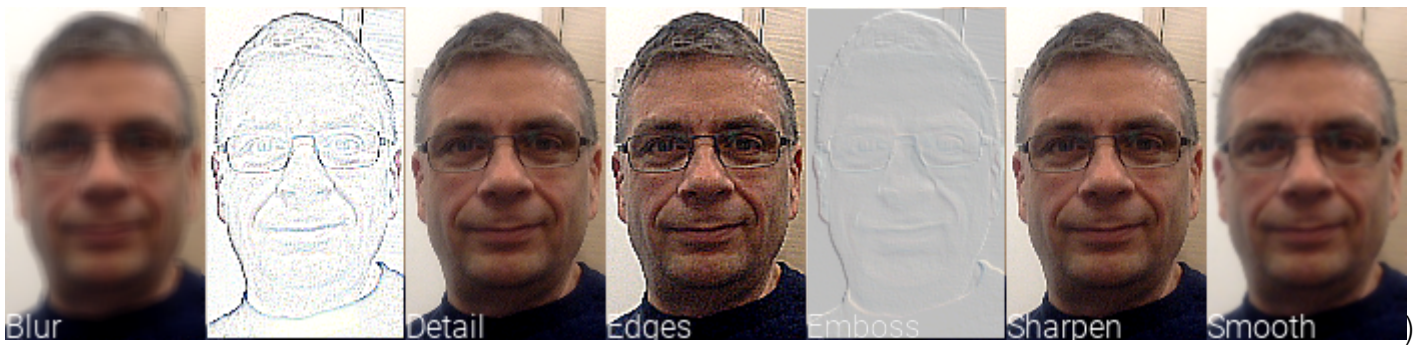
### Apply an image filter

There are seven types of filters built in to the Python Image object. They are

- ImageFilter.BLUR
- ImageFilter.CONTOUR
- ImageFilter.DETAIL
- ImageFilter.EDGE\_ENHANCE
- ImageFilter.EMBOSS
- ImageFilter.SHARPEN
- ImageFilter.SMOOTH

The following is an illustration of the effect provided by each...





An example of using the ImageFilter to blue is provided below. To use one of the other filters, simply replace the `ImageFilter.BLUR` with the relevant filter you wish to use.

```
img = Image.open("my picture.png") # alternatively use the camera
new_img = img.filter(ImageFilter.BLUR)
new_img.show()
```

## Apply an Image Enhancer

Similar but separate to the ImageFilters are also Image Enhancers. These are used to adjust the brightness, contrast, sharpness or color saturation within an image.

The following is an illustration of the effect provided by each...



To increase the brightness of an image, we supply a number above `1.0` to the enhance function, to decrease the brightness we supply a number below `1.0` to the enhance function. You can go as high or low as you like.

```
# Increase brightness
img = Image.open("my picture.png") # alternatively use the camera
new_img = ImageEnhance.Brightness(img).enhance(1.5)
new_img.show()

# Decrease brightness
img = Image.open("my picture.png") # alternatively use the camera
new_img = ImageEnhance.Brightness(img).enhance(0.5)
new_img.show()
```

To enhance the contrast, the key line would be...

```
new_img = ImageEnhance.Contrast(img).enhance(0.5)
```

To enhance the sharpness use...

```
new_img = ImageEnhance.Sharpness(img).enhance(0.5)
```

Finally, to enhance colour saturation, use...

```
new_img = ImageEnhance.Color(img).enhance(0.5)
```

## Your task/s

- What are the limits to the various filters and enhancers?
- What happens if you reuse a filter on an already filtered photo? eg: Applying blur on an already blurred image? Compare the before, after-first and after-second photos.

## 4. Create a meme

---

To create a meme you usually need a couple of things: A great photo or image, and a witty caption.

I'm fortunate enough that one of the students at my previous school created a meme about me that wasn't of the 'this teacher sucks' variety, so I'll happily share it here as a sample...



We've covered how you can manipulate photos and images, but what can we do about the caption?

Python helps us out here by providing tools to draw and add text to our images.

Caution: If you are going to create a meme featuring someone else in the school - get their permission first (celebrates are fair game though... but remember this is a school assignment, and school expectations on appropriateness apply)

The following section requires the following imports...

```
from PIL import Image, ImageDraw, ImageFilter, ImageEnhance, ImageFont
import ImageTools
```

### Draw a line on an image

To do any drawing with the Image tools, we need to crate an **ImageDraw.Draw** object that is linked to our image. That is step 1 in the code below.

The **draw.line()** function requires the two sets of x,y coordinates representing the start and end point for the line and a fill color. Optionally you can include a width (if you leave it out it will default to 1 pixel wide). This is step 2 in the sample code.

```
camera = ImageTools.Camera()
img = camera.take_photo()
# Step 1 - Create a draw object that is linked to the img
draw = ImageDraw.Draw(img)
# Step 2 - Use the draw object's line function
```

```
x1,y1=100,150
x2,y2=400,300
draw.line((x1,y1,x2,y2), fill=(255,255,0), width=1)
img.show()
```

Other than mems, drawing lines can also have other uses. Here is one I created that draws gridlines on an image - a helpful way of finding coordinates for different features I might want to `.crop()` or replace with a `.paste()`. Every 100 pixels this draws a thick yellow line, and every 20 pixels a thin yellow line.

```
filename = input("What image do you want to open?")
img = Image.open(filename)
width,height = img.size
print(f"Image size is {width} x {height} pixels")
draw = ImageDraw.Draw(img)
for x in range(width):
    if x % 100 == 0:
        draw.line((x,0,x,height), fill=(255,255,0), width=3)
    elif x % 20 == 0:
        draw.line((x,0,x,height), fill=(255,255,0), width=1)
for y in range(height):
    if y % 100 == 0:
        draw.line((0,y,width,y), fill=(255,255,0), width=3)
    elif y % 20 == 0:
        draw.line((0,y,width,y), fill=(255,255,0), width=1)
img.show()
```

## Draw a rectangle on an image

Drawing a rectangle is very similar to drawing a line. Again we provide two sets of coordinates, effectively representing the top-left corner and the bottom-right corner of the rectangle. (if you want a rectangle that is not parallel with the edge you'll have to use rotate on it later).

The following example will draw four rectangles around the outer edge of our photo, with the effect of creating a black border around our photo that is 50 pixels wide.

```
camera = ImageTools.Camera()
img = camera.take_photo()
width, height = img.size
# Step 1 - Create a draw object that is linked to the img
draw = ImageDraw.Draw(img)
# Step 2 - Use the draw object's rectangle function
draw.rectangle((0,0,width,50), fill="#000000", width=1)
draw.rectangle((0,0,50,height), fill="#000000", width=1)
draw.rectangle((width-50,0,width,height), fill="#000000", width=1)
draw.rectangle((0,height-50,width,height), fill="#000000", width=1)
img.show()
```

## Draw an ellipse on an image

The ellipse function works by providing it the coordinates of a rectangle, `[left, top, right, bottom]`, and it will then draw an ellipse that touches all the sides of the rectangle. For example, to draw an ellipse that is the size of our image...

```
camera = ImageTools.Camera()
img = camera.take_photo()
width, height = img.size
# Step 1 - Create a draw object that is linked to the img
```

```
draw = ImageDraw.Draw(img)
# Step 2 - Use the draw object's rectangle function
draw.ellipse((0,0,width,height), outline="#ffff00", width=5)
img.show()
```

## Change an individual pixel

What is the point of being able to read or update an individual pixel? Because where there is one, there is many. Knowing how to modify one gives you the capacity to run the functionality through some loops and change a whole bunch of pixels. It is the most fine-detail level of control you can have.

To read the colour values for an individual pixel...

```
camera = ImageTools.Camera()
img = camera.take_photo()
width, height = img.size
for y in range(height):
    for x in range(width):
        col = img.getpixel((x,y))
        print(f"The colour at {x},{y} is {col}")
```

To set the colour values for an individual pixel...

- use the `Image.putpixel((x,y), colour_code)` function for example...

```
img = Image.new("HSV", (255, 255))
width, height = img.size
for x in range(width):
    for y in range(height):
        img.putpixel((x,y), (x,y,255))
img.show()
```

## Write text on an image

Firstly to draw text on your Image you are going to need to pick a font. I recommend using <https://fonts.google.com/>, find one you like (that's free) and download the font. It will probably download as a ZIP file. Viewing the file icon in Windows Explorer, right click on it and find the option to "decompress" or "expand" so you get the TTF file(s). Copy your TTF files into your project folder in order to proceed.

In addition to creating an `ImageDraw.Draw()` object, we also need to create an `ImageFont.truetype()` object which will contain the font information from the TTF file we just downloaded.

The `draw.text()` command brings it all together. It requires four parameters: the location as (x,y) pixel coordinates for where to place the top left of the text, the string containing the caption text you wish to write, a colour value, and a link to the font object.

See this for a working example...

```
camera = ImageTools.Camera()
img = camera.take_photo()
w,h = img.size
person = input("Who is in this photo?")
# Generate the text we will place on the Image
caption = "This is a photo of "+person
```

```
# Nominate a colour
yellow = "#ffff00"
# Determine the (x,y) location to place the text
location = (20, h-50)
# Create a drawing object linked to our image
draw = ImageDraw.Draw(img)
# Create a drawing object linked to our image
font = ImageFont.truetype("Roboto-Light.ttf", 48)
# Finally, bring it all together and render the text to our Image
draw.text(location, caption, yellow, font=font)
img.show()
```

Note: If you wish to centre or right align your text, check the hint suggested here  
<https://stackoverflow.com/a/1970930/10971929>

## Your task

Create a meme! Combine a photo, maybe put a border around it, add some text, go viral!

Upload your meme to your portfolio.

Reminder: If you are going to create a meme featuring someone else in the school - get their permission first (celebrates are fair game though... but remember this is a school assignment, and school expectations on appropriateness apply)

## 5. Object detection

---

The following section requires the following imports

```
from PIL import Image, ImageDraw, ImageFilter, ImageEnhance
from PIL import ImageFont
import ImageTools
```

### Detect a face

The algorithms involved to detect a face are actually not very complex (relatively speaking). It doesn't even require what would properly be considered as artificial intelligence. Faces (and other features) can be detected based on mathematical pattern recognition. Fortunately for you, this is a task so commonly used, you don't need to know any of the maths involved, you can just use an existing library. The ImageTools library you are using will do this for you (which is then using another library called OpenCV).

In fact it is now simply a case of needing the `ImageTools.get_faces()` command as shown...

```
camera = ImageTools.Camera()
img = camera.take_photo()
faces = ImageTools.get_faces(img, "haarcascade_frontalface_default.xml")
print(faces)
```

The second parameter, `"haarcascade_frontalface_default.xml"`, is the name to a file that must be in our project folder. It is the file that contains the training data of what a face looks like. The function will load this file and use the information in it to determine if there is a face in the Image you supply. You can download this file here...

- <https://github.com/opencv/opencv/tree/master/data/haarcascades>

When you run this, assuming the image contains a face, the print command will output a set of numbers like this...

```
[[537 183 321 321]]
```

You should hopefully recognise this as a Python list. More correctly it is a list of lists (note the double sets of square brackets). This is the top-left-x-coordinate, the top-left-y-coordinate, width, and height for a face found in the Image.

If there is more than one face, you will get one set of coordinate information for each face. Such as this which came from a photo with three people in it...

```
[[ 280  313  208  208]
 [1354  330  217  217]
 [ 674  454  196  196]]
```

Once we have the coordinate information we can then use the crop tool to create new Images containing just the faces...



```
counter = 0
camera = ImageTools.Camera()
img = camera.take_photo()
faces = ImageTools.get_faces(img, "haarcascade_frontalface_default.xml")
# for each individual face in the list of faces...
for a_face in faces:
    # extract the left, top, width and height locations of a face
    x,y,w,h = a_face
    a_face_img = img.crop((x,y,x+w,y+h))
    a_face_img.save(f"face_{counter:2}.jpg", "jpg")
    a_face_img.show()
    counter = counter + 1
```

Or we could use a drawing tool to put rectangles highlighting the faces found in the original image...

```
camera = ImageTools.Camera()
img = camera.take_photo()
draw = ImageDraw.Draw(img) # create the drawing object
faces = ImageTools.get_faces(img, "haarcascade_frontalface_default.xml")
# for each individual face in the list of faces...
for a_face in faces:
    # extract the left, top, width and height locations of a face
    x,y,w,h = a_face
    # draw a rectangle around the face
    draw.rectangle((x,y,x+w,y+h), outline="#ffff00", width=5)
# show the final image, highlighting each face
img.show()
```

Or, we can use it to have some fun such as making an Instagram or Snap style face-filter (next lesson).

## Your task/s

- What range of angles and lighting/shadow will work?
- What is the minimum size a face needs to be to be detected?
- How many faces at once can it reliably detect? (ensure each face is over the minimum size for this to be reliable)
- Is there any bias in this algorithm? Experiment with different gender, racial and ethnic faces. Does this algorithm have weaknesses, and if so in what way? (see note below)

Note regarding bias: The algorithm we are using is one that is very widely used for facial recognition systems around the world. But pattern recognition algorithms, such as this one, are not perfect. They are only as good as the programmers that wrote them, and the programmers used to build it. For instance some facial recognition algorithms have been known to show bias in being more reliable at detecting people with lighter skin instead of darker skin. This creates enormous challenges for equity and equality when even the "neutral" computer can be biased.



## 6. Complete the face filter demo



We now have all the pieces necessary for an Instagram/Snap style face filter. In short we want to

- Take a photo
- Load our "face filter" image containing the cartoon eyes/nose/ears/whatever
- Detect any faces in the photo
- Obtain the location for all the faces in the photo
- Paste our "face filter" image over the top of each face in the photo (resizing it to match each face first)
- Done! Sounds easy enough right?!

```
# Take a photo
camera = ImageTools.Camera()
img = camera.take_photo()
# Load our "face filter" image containing the cartoon eyes/nose/ears/whatever
face_filter = Image.open("face-filter-demo.png")
# Detect any faces in the photo
faces_coordinates = ImageTools.get_faces(img, "haarcascade_frontalface_default.xml")
print(faces_found)
# For all the faces detected in the image....
for a_face in faces_found:
    # Obtain the location for this individual face
    x,y,w,h = a_face
    # Resize the face filter to match this face
    resized_face_filter = face_filter.resize((w,h))
    # Paste the face filter into the main photo
    img.paste(resized_face_filter, (x,y))
# Show and save the end result
img.show()
img.save("masterpeice.png", "png")
```

You've probably discovered your first attempt is not perfect. It will need some tweaking. You may have to adjust the coordinates for the paste, or adjust the amount of the resize.

Be aware that any calculations made to adjust your numbers should still end up with whole integers. For instance if you took a size of (232,135) and decided to stretch the height by a factor of 1.2, that would give you decimals but part pixels don't exist. So you would have to use the `int()` function like `.resize((w, int(h*1.2)))`

## Your task/s

Use an existing PNG with animal cartoon features before attempting to create your own. I have a collection of a few available here:

- <https://github.com/paulbaumgarten/paulbaumgarten/tree/master/myp-design/image-editing/filters>

Alternatively, use a Google Image search with terms such as "face filter png transparent"

Upload your code, before photo and after photo to your portfolio.

## 7. Design your own effects, memes and filters

---

Enough of following the examples, time to build something of your own from scratch.

### Your task/s

- You should design a two or three different programs. Your goal is to apply a range of the techniques we have learnt in this unit. At the basic level this could include crop, resize, paste, and the built in filter and enhance functions. You should also have a program that uses the drawing functions, and finally a program to use the face recognition functions.
- Draw accurate mock ups of what you'd like the before and after images to resemble.
- Add annotation to your mock ups to describe which method you are planning to use.
- Scan/photograph your mockups and upload them to your portfolio for this unit.

## 8. Build your own effects, memes and filters

---

### Your task/s

- Use Photoshop, Illustrator or any other preferred tool to create your blank face filter images.
- Use Python to build your photo effect and filter programs that you designed.
- Tweak to optimise it as best as possible
- Test your filters on a variety of people
- You should have your programs save the before and after of key sample photos. Upload the relevant before and after photos to your portfolio.
- Upload the code for each tool you created to your portfolio.

### Pro-tip

It is suggested to use code that will automatically generate filenames that are unique, so you can take lots of photos when testing without worrying about losing any (of course nothing to stop you deleting some later).

The following will use the date and time to create a unique filename (provided you don't take more than one photo per second).

Note: that it requires you to add `from datetime import datetime` to your import statements. An example follows...

```
# Create a unique filename string based on the date and time for example 20191223-202613
filename = datetime.now().strftime("%Y%m%d-%H%m%S")
# Create the camera object
camera = ImageTools.Camera()
# Take 100 photos
for i in range(100):
    # Take a photo
    img = camera.take_photo()
    # Do something interesting with the photo, such as convert to black and white
    bw = img.convert(mode="1")
    # Save both images using the special filename prefix
    img.save(filename+"-before.png", "png")
    bw.save(filename+"-after.png", "png")
```

## 9. Evaluate your effects, memes and filters

---

How successful were you? Address the following questions and provide the response to your portfolio.

### Your task/s

- Task 1: How did you test each photo effect/filter program you wrote? Your answer should include:
  - How you ensured it functioned as a basic program without generating errors
  - How it worked on a range of photos
- Task 2: How successful were you with your photo effect/filter programs? Your answer should include:
  - In which cases did it work well, in which cases did it not work ideally?
  - What thoughts do you have as to what caused the difference in each case?
- Task 3: Given more time, what changes would you make? Why?
- Task 4: Through tools such as Instagram and Snapchat there is now an abundance of easily accessible filters such as the ones we have created. Briefly comment on how you think this is affecting society. Is it potentially damaging to individual self-worth or societal harmony, or is that just fear mongering? Justify your opinion.

# References

---

1. Color modes from <https://pillow.readthedocs.io/en/5.1.x/handbook/concepts.html>
2. Haar cascades from <https://github.com/opencv/opencv/tree/master/data/haarcascades>