

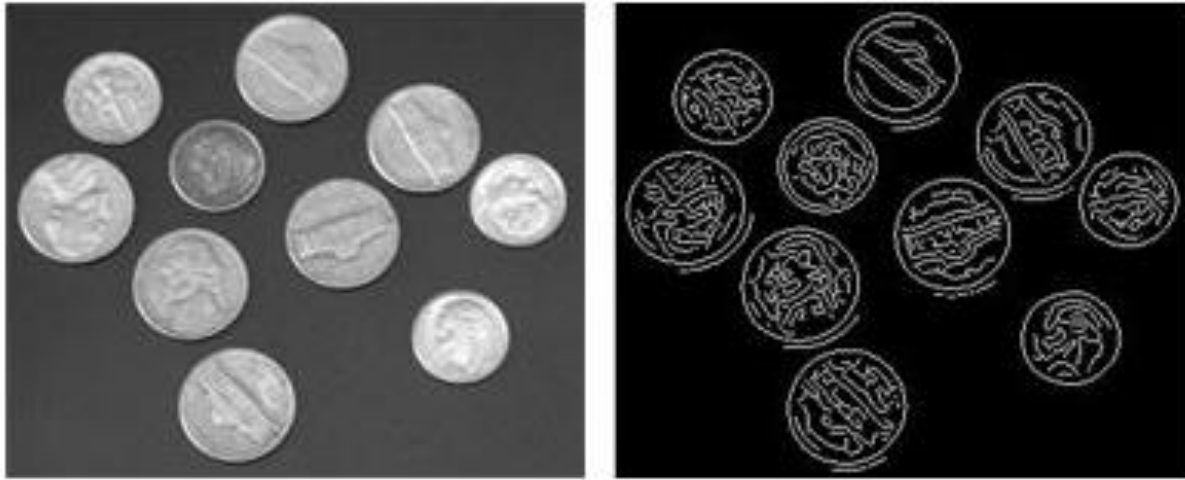


Image segmentation using the Canny method.

Source: <https://www.mathworks.com/discovery/edge-detection.html>

CMPT 120

Lecture 25 – Computer Vision

Python – Images files, tuples
and nested **for** loops

Last Lecture

- Started investigating another field in Computing Science: **Computer Vision!**
- Had another look at **Lists!**
- Introduced **file IO** -> **I**nput/**O**utput
 - **text file**

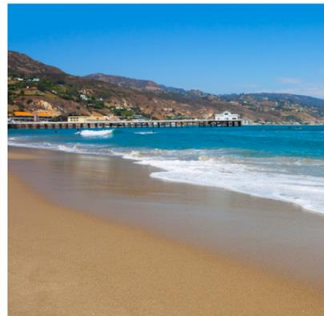
Today's Menu

- Solve the **image processing** problem of merging one image onto another
- And in the process, have a closer look at
 - Using the **PIL** module
 - How to open an **image file**
 - How to read the content of an **image file**
 - **Pixels**
 - **RGB** colour scheme
 - How to merge two images together
 - Nested for loops
 - Tuples

Let's give a go!

Step 1 - Problem Statement

- Combine (merge) the image file **kid-green.jpg** with the image file **beach.jpg** such as to produce an image file that displays the kid on the beach!



Let's get started!

```
# CombinedImages.py
#
# Description: Take two images (2 files) as input
#              1) Image A is the foreground object in front of a green screen
#              2) Image B is the background
#              and combine them.
#              Once combined, the resulting image has the foreground object in
#              front of the background.
#
# Author: AL and AL
# Date: 2023

# Import necessary Python Imaging Library (PIL)
from PIL import Image
```

Image is a module inside the **Python Imaging Library (PIL)** package, which is a collection of modules.

The **Image** module contains pre-built functions that will help us do image processing.

Opening Image Files

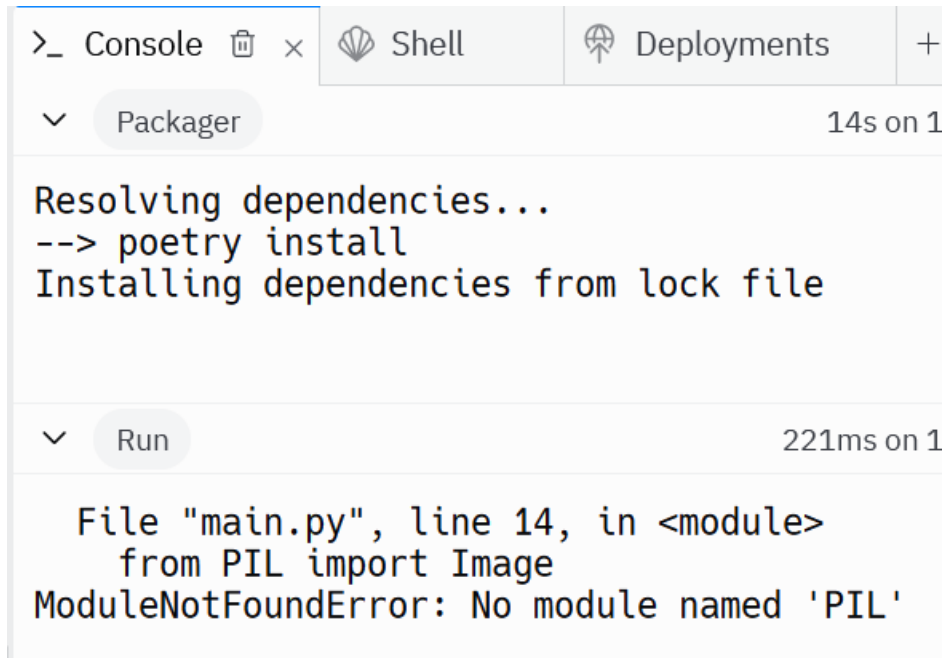
```
# CombinedImages.py
#
# Description: Take two images (2 files) as input
#              1) Image A is the foreground object in front of a green screen
#              2) Image B is the background
#              and combine them.
#              Once combined, the resulting image has the foreground object in
#              front of the background.
#
# Author: AL and AL
# Date: 2023

# Import necessary Python Imaging Library (PIL)
from PIL import Image

# Open the file containing the foreground object with green screen -> A
imageKidOpen = Image.open("kid-green.jpg")

# Get width and height of image kid-green.jpg
widthKid = imageKidOpen.width
heightKid = imageKidOpen.height
print(f'Image kid-green.jpg is {widthKid} by {heightKid}')
```

Problem running?



The screenshot shows a terminal window with three tabs: Console, Shell, and Deployments. The Console tab is active and shows the output of a poetry install command. The output indicates that dependencies are being resolved and installed from a lock file. Below this, the 'Run' section shows a Python error: a ModuleNotFoundError for the 'PIL' module in 'main.py' at line 14.

```
>_ Console [x] [Shell] [Deployments] +  
  v Packager 14s on 1  
Resolving dependencies...  
--> poetry install  
Installing dependencies from lock file  
  
  v Run 221ms on 1  
File "main.py", line 14, in <module>  
    from PIL import Image  
ModuleNotFoundError: No module named 'PIL'
```

Solution

- Click the Shell tab  Shell
- And enter: **upm add Pillow**

```
~/IllSomePasswords$ upm add Pillow  
--> poetry add Pillow  
Using version ^10.2.0 for pillow
```

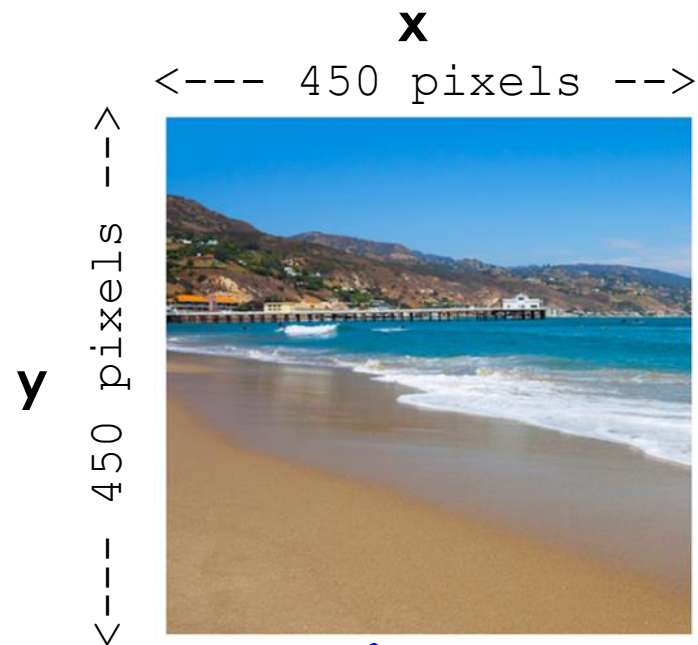
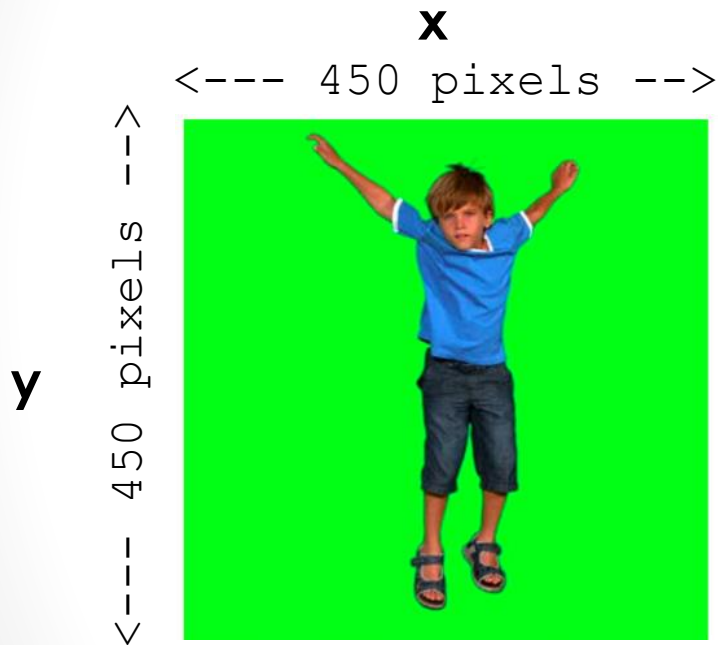
```
Updating dependencies  
Resolving dependencies... (0.2s)
```

```
Package operations: 1 install, 0 updates, 0 removals
```

- Installing pillow (10.2.0)

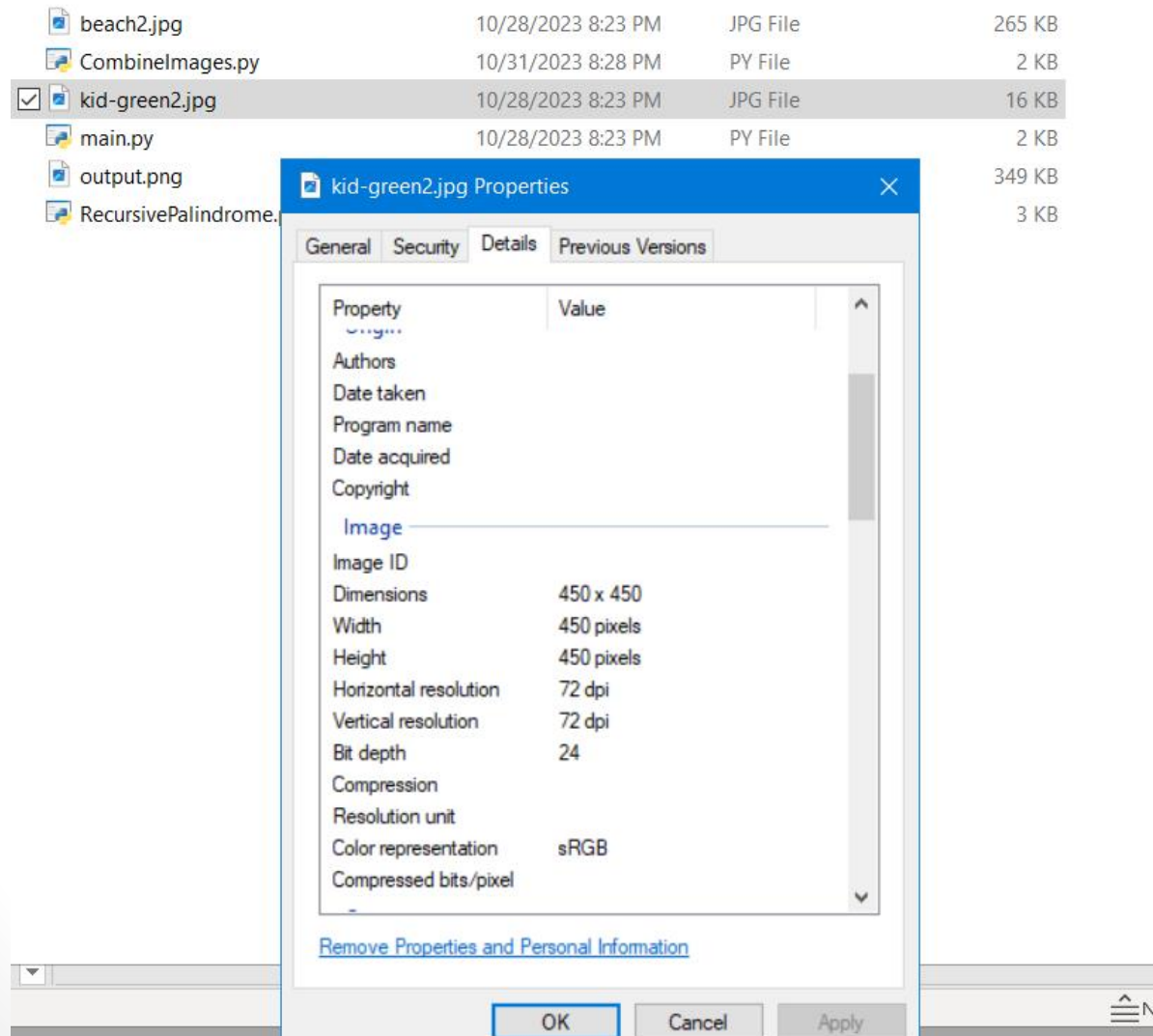
```
Writing lock file
```

Image files



For the sake of simplicity,
let's use the same
sized image here

File Properties on Windows



open() versus load()

```
# CombinedImages.py
#
# Description: Take two images (2 files) as input
#              1) Image A is the foreground object in front of a green screen
#              2) Image B is the background
#              and combine them.
#              Once combined, the resulting image has the foreground object in front
#              of the background.
#
# Author: AL and AL
# Date: 2023
```

```
# Import necessary Python Imaging Library (PIL)
from PIL import Image

# Open the file containing the foreground object
imageKidOpen = Image.open("kid-green.jpg")

# Get width and height of image kid-green.jpg
widthKid = imageKidOpen.width
heightKid = imageKidOpen.height
print(f'Image kid-green.jpg is {widthKid} by {heightKid}')

# Load the content (pixels) of the file containing the
# foreground object with green screen -> A
imageKidLoaded = imageKidOpen.load()

# Open the file containing the background -> B
imageBeachLoaded = Image.open("beach.jpg").load()
```

open() gives you access to metadata information about the image such as its width, height, etc.

load() can be run once **open()** has been executed. **load()** gives you access to the pixel values.

Load Image Files

```
# CombinedImages.py
#
# Description: Take two images (2 files) as input
#              1) Image A is the foreground object in front of a green screen
#              2) Image B is the background
#              and combine them.
#              Once combined, the resulting image has the foreground object in
#              front of the background.
#
# Author: AL and AL
# Date: 2023

# Import necessary Python Imaging Library (PIL)
from PIL import Image

# Open the file containing the foreground object with green screen -> A
imageKidOpen = Image.open("kid-green.jpg")

# Get width and height of image kid-green.jpg
widthKid = imageKidOpen.width
heightKid = imageKidOpen.height
print(f'Image kid-green.jpg is {widthKid} by {heightKid}')

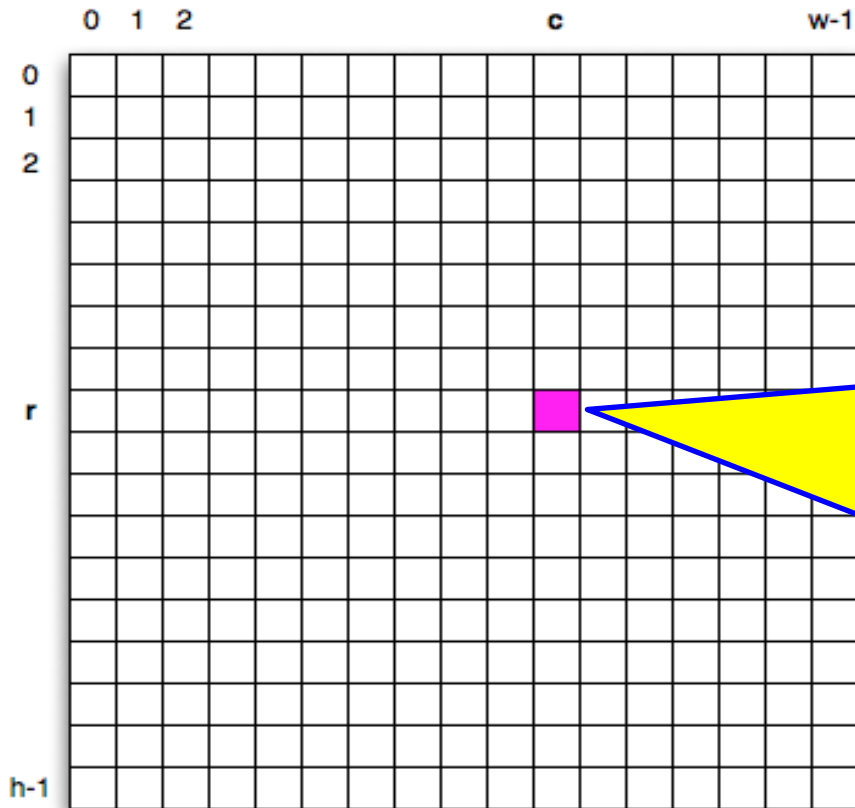
# Load the content (pixels) of the file containing the
# foreground object with green screen -> A
imageKidLoaded = imageKidOpen.load()

# Open the file containing the background -> B
imageBeachLoaded = Image.open("beach.jpg").load()
```

What do image
files contain?

Image files contain pixels!

- **pixel**: picture element
- How do we access each pixel?



This is a 2-dimensional (**2D**) **table** containing our image (in other programming languages, sometimes it is called a 2D **array**)

We can access the value of this 2D table element (i.e., pixel) the same way we access the elements of a list i.e., using indices, one index per dimension:
`my_image[c, r]`
e.g. `imageBeach[10, 8]`

What do pixels contain?

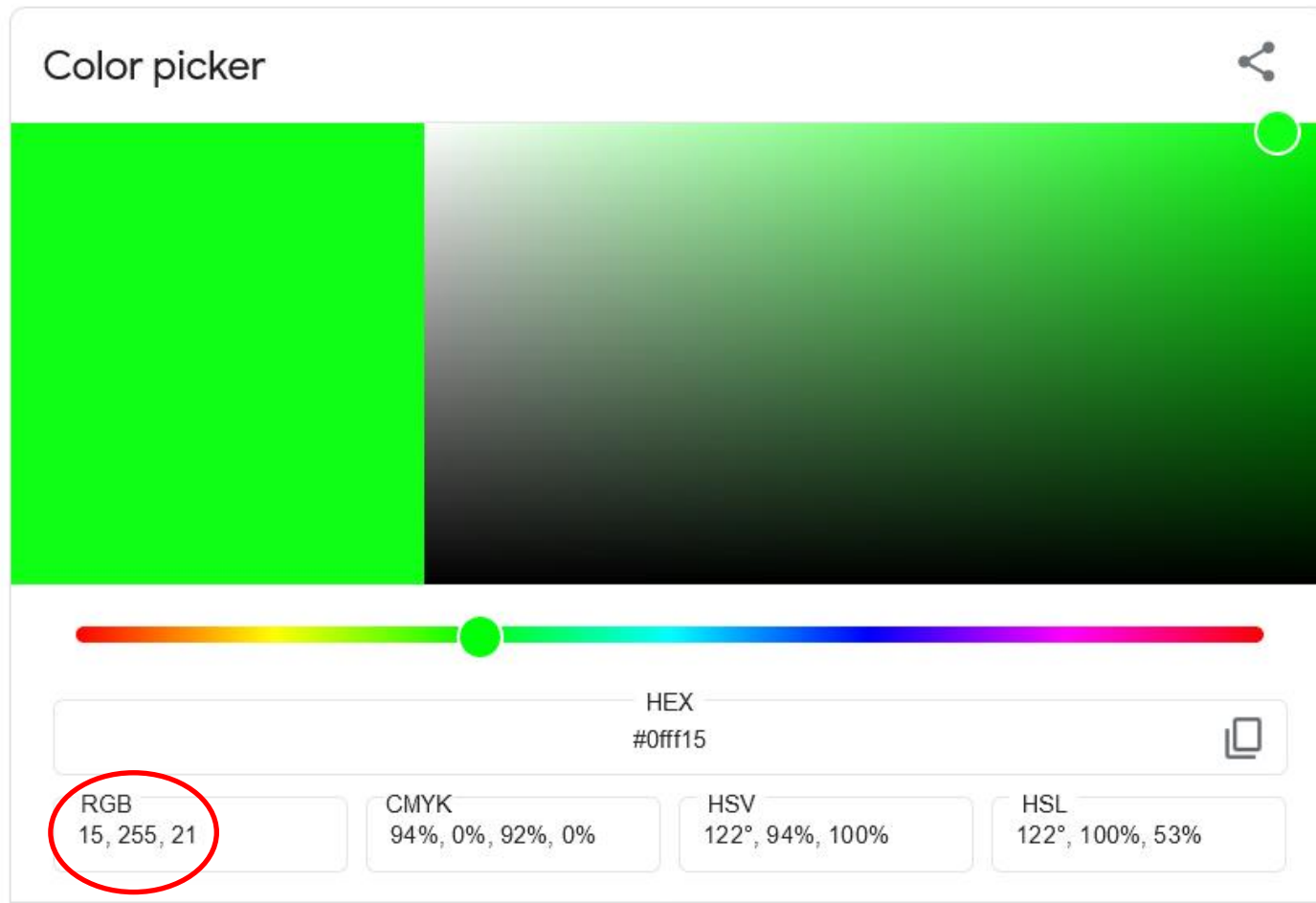
Testing ... Testing ...

- What are pixels made of?
- **Step 1 – Problem Statement**
 - Print the value found of coordinates (0,0) of the image file `kid-green.jpg`?

Review - Tuples

- Like a list, but immutable
- Tuples are like ***read-only*** lists

Color Picker



Source: <https://www.google.com/search?client=firefox-b-e&q=colourpicker>

How to know the colour of a pixel?

255 is the maximum intensity

Color	Red	Green	Blue
Red	255	0	0
Green	0	255	0
Blue	0	0	255
White	255	255	255
Black	0	0	0
Yellow	255	255	0
Magenta	255	0	255

Back to our “combining images” problem ...

Step 2 – Design

- Let's have a look at the rest of the comments in the **CombineImages.py** program

Step 3 – Implementation

- Let's translate these comments into Python code keeping in mind the following questions:
 1. How to go through all pixels of A?
 2. How to find the corresponding pixel in B?

Next lectures

- **Practice Exam #7**
- Continue having fun processing images
- How to create modules
- We shall talk about our Assignment 4 and our Assignment 5