Welcome to Week 10

Press Space to start → Photo by Kseniia Rastvorova on Unsplash

Agenda for Today

- Housekeeping
- Content: Modules and Image Processing!

Housekeeping

- things that are due for the rest of the semester
  - assignment 5 (nov 10)
  - project part 1 (listed as nov 15, but will be extended to Nov 22.)
    - I posted the Canvas page for this just so you can see the deadline; full details will be posted after lecture.
  - project part 2 (listed as dec 1, cannot extend this one!)
  - you need to show up to the final in person and turn your exam (very important!)

That's it! I'll keep all these things in mind for our remaining class time and try to help you make progress on project and exam prep.

Overview of content and reminder of learning outcomes (slide 1)

You should know:

- Modules!
  - How to create and use a module containing one's own defined functions
  - How to import and access the contents of packages and modules
  - How to import a module with a short name

Overview of content and reminder of learning outcomes (slide 2)

- Image processing and 2D list / 3D list manipulation:
  - How pixel colors are represented by RGB values
  - How to pass global variables into local scope
  - That you can to directly return a Boolean expression in a function without using if statement in the function (e.g. `return x > 10`)
  - How to access and modify a 2D image in the form of a list of lists (with 0s and 1s)
  - How to access and modify a 3d image in the form of a list of lists (with 3-item RGB lists)

- More from Image processing and 2D list / 3D list manipulation:
  - How to manipulate 2 dimensional lists, process a row, process a column, process a specific element
  - How to read, show and save images using the 3Dlist representing an image as provided in the cmpt120images module
  - How to extract and/or change the color as RGB and/or individual color components of a pixel using a 3DList representing an image as provided in the cmpt120images module

New Module: Broad Intro to Computer Vision

Many applications:

- captioning photos
- facial recognition (unlock your phone)
- filters to make your photos look cool
- medical applications
- lots of experts at SFU!

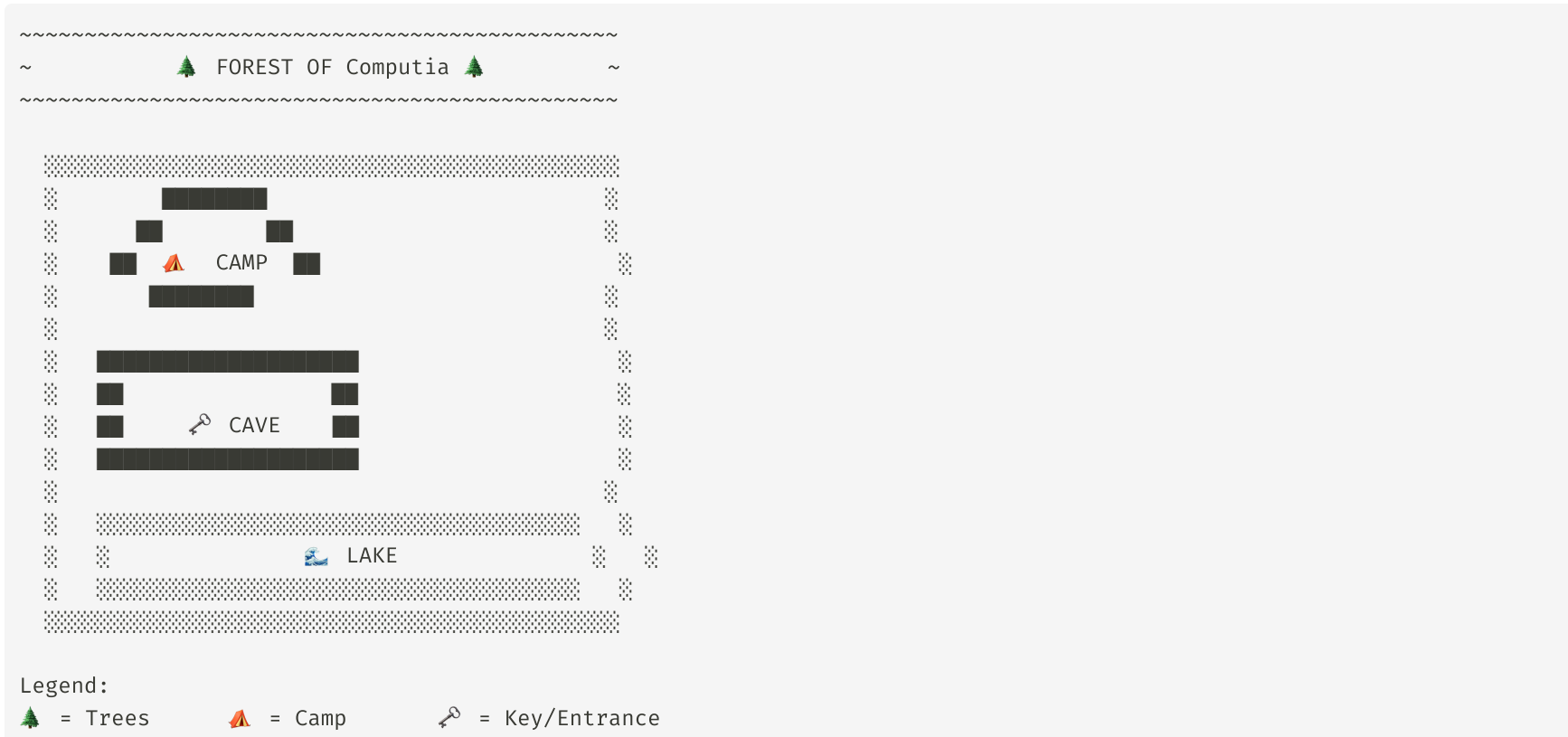
What is an image?

Simple question: "What is an image?""

- Philosophically?
- In analog media?
- In digital media?

A "Map" in Terminal

What if we wanted to make a game in our terminal?



A "Map" in Terminal

Using mostly zeros and 1s?

```
1111111111111111111111111111111111111111111111
1000000000000000000000000000000000000000000001
1000011111111110000000000000000000000000000001
1000110000000011000000000000000000000000000001
1001100  CAMP  0011000000000000000000000000001
1000100111111000100000000000000000000000000001
1000000000000000000000000000000000000000000001
1000000000000000000000000000000000000000000001
1000111111111111111110000000000000000000000001
1000111000000000001110000000000000000000000001
1000111   CAVE   1110000000000000000000000001
1000111000000000001110000000000000000000000001
1000111111111111111110000000000000000000000001

Legend:
1 = Wall/Boundary
0 = Open path
CAMP and CAVE are marked as labeled landmarks within the map.
```

As actual python code

```
map_grid = [
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
    [1, 0, 0, 0, 0, 1, 0, 0, 0, 1],
    [1, 0, 1, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 0, 1, 0, 0, 0, 1, 0, 1],
    [1, 1, 0, 1, 1, 1, 0, 1, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 1, 0, 1],
    [1, 0, 1, 1, 1, 1, 0, 1, 0, 1],
    [1, 0, 0, 0, 0, 1, 0, 0, 2, 1],
    [1, 1, 1, 1, 0, 1, 1, 1, 1, 1],
]

# Display the map
for row in map_grid:
    print(" ".join(str(cell) for cell in row))
```

As Python code, but RBG triplets

```
map_grid_rgb = [
    [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
    [[0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [0, 0, 0], [255, 255, 255], [255, 25
    [[0, 0, 0], [255, 255, 255], [0, 0, 0], [0, 0, 0], [255, 255, 255], [0, 0, 0], [255, 255, 255], [0, 0, 0], [255, 255
    [[0, 0, 0], [255, 255, 255], [255, 255, 255], [0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255], [0, 0, 0
    [[0, 0, 0], [0, 0, 0], [255, 255, 255], [0, 0, 0], [0, 0, 0], [0, 0, 0], [255, 255, 255], [0, 0, 0], [255, 255, 255
    [[0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [0
    [[0, 0, 0], [255, 255, 255], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [255, 255, 255], [0, 0, 0], [255, 255, 255
    [[0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [0, 0, 0], [255, 255, 255], [255, 25
    [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [255, 255, 255], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]
    [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
]

# Display the map in a human-readable way
for row in map_grid_rgb:
    print(" ".join(str(cell) for cell in row))
```

Ok, this is obviously getting a bit ridiculous.

If you really had to, I'd bet you could "complete this maze"…

but let's actually visualize it!

Motivating Task: Green Screen

Today's goal:

write a program that can merge a green screen image with another background, using the following concepts:

- Colours in RGB space
- Pixel representations of images in 2D arrays, e.g. `image[row][col]`
- Functions that return values
- Modules

Green Screen

What's a green screen?

https://en.wikipedia.org/wiki/Chroma_key

Prerequisites: Downloading images

We need to to download some files.

- `cmpt120image.py`
- `kid-green.jpg`
- `beach.jpg`

`cmpt120image.py` is in GitHub > other_content > week10_more_images

(github.com/nickmvincent/cmpt120_fall2024)

At this link

The images in this file are for an extra challenge

Images in Canvas > Modules > Week 10.

Prerequisites: Installing a new Python library

And we need to install a new Python library, `pygame`

Using cmpt120image.py and pygame

Ok, time to import an existing image.

- Download pygame:
  - Try: `pip3 install pygame`
    - Didn't work? try `pip install pygame`
      - Didn't work? try `python3 -m pip install -U pygame==2.6.0`
        - Didn't work? Office hours

3D List

We're going to use the list-of-lists-of-3-item-list approach we saw above

(In Learning Outcomes, this is called 3D List)

- outermost list = rows
- second layer list = columns
- innermost list = RGB value

```
image = [
    [ [0,0,0]  [255,255,255] ]
    [ [0,0,0], [255,255,255] ]
]
```

RGB reminders

Important (testable!) facts about RGB

- 3 integers between 0 and 255
- requires 2 digits in hexadecimal (between 0 - F)
- A is 10, B is 11, C is 12, D is 13, E is 14, F is 15
- only 16 options in hexadecimal so we just need 4 bits per character

```
FF → 15 15 → 1111 1111
5A → 5 10 → 0011 0110
```

One single color (e.g. "light red") thus uses up 4 * 2 * 3 = 24 bits

Conventions about rows and columns (120 and beyond)

- Should row come first or column come first? It's just convention.
- in other words: just another choice someone made in implementing a language / software / discipline!
- like the default arguments of our Python methods, the choice of which Python keywords are "special"
- Somebody just liked the words "for", "if", etc.
- In linear algebra $A_{i,j}$ conventionally means the i-th row, j-th column of matrix A

Example

$A_{1,1}, A_{1,2}, A_{1,3}$

$A_{2,1}, A_{2,2}, A_{2,3}$

$A_{3,1}, A_{3,2}, A_{3,3}$

See e.g.

https://en.wikipedia.org/wiki/Matrix_(mathematics)

But compare with

https://en.wikipedia.org/wiki/Cartesian_coordinate_system

Where <1,2> means go 1 unit right on the x-axis, then 2 units up.

Our convention:

- start from the top left
- first index for rows
- second index for columns
- Start with ZERO

This is a nice convention because it maps to how we loop through a matrix X, say: 1 2 3 4

```
X = [[1,2], [3,4]]

for row in range(len(X)):
  for column in range(len(X[0])):
    print(X[row][column], end='')
  print()
```

Write the indices!

Just like string indexing problems, if you're worried about making a mistake on paper questions just write the indicies!

```
A =
1 2
3 4

    0 1
  0 1 2
  1 3 4
```

Learn these conventions, but prepare for others

But ultimately, we are just somewhat-arbitrarily choosing to say

“Let’s represent our image as a list of lists… Each of the individual lists will correspond to a row”

We could have each list represent a column. We could use one-indexing. Etc!

Why harp on this?

- For everyone: help avoid confusion and errors with rows and columns
- For those who want to do more linear algebra, computer vision, etc. – it may be of interest!

Height and width

Height and width of of an image use this list-of-lists (black and white) or list-of-list-of-lists (RGB) approach?

How to get height?

How to get width?

Try it!

- Right now, try a quick test in your terminal
- First, define a grid / matrix of integer values, using a list of lists
  - E.g., `image = [[1,2], [3,4]]`
- Then, try access each of the individual values
  - `image[0][0], image[0][1]`
- Then, try editing one of individual values

Starter code for green screen

In your new script, here's our starting code:

```
import cmpt120image as img_module
img_module.init()

img = cmpt120image.get_image('chatgpt_raccoon.jpg')
print(img[0][0])
```

A single pixel?

What's in `img[0][0]`?

If time: Brief intermission, look at some example RGB color values.

Reminder of our goal:

Write a program that can merge a green screen image with another background, using the following concepts:

- Colours in RGB space
- Pixel representations of images in 2D arrays, e.g. `image[row][col]`
- Functions that return values
- Modules

How to merge a green screen image?

To merge a green screen image, we probably need to do something with all the green pixels?

What function might we want, to start?

is_green function

Complete the `is_green` function in Python.

It should return True if the red, green and blue channels are all within 30 of 0, 255, and 0 respectively.

```
def is_green(r,g,b):
    """
    Return true if the rgb value are within 30 of the color <0,255,0>
      in other words, the r value can be 0-30, the g value 225-255, and the b value 0-30
    input:
      r - the red value (0-255)
      g - the green value (0-255)
      b - the blue value (0-255)
    return True or False
    """
```

```
def is_green(r,g,b):
    """
    Return true if the rgb value are within 30 of the color <0,255,0>
      in other words, the r value can be 0-30, the g value 225-255, and the b value 0-30
    input:
      r - the red value (0-255)
      g - the green value (0-255)
      b - the blue value (0-255)
    return True or False
    """
    return r < 30 and g > 255 and b < 30
```

Note that

```
return r < 30 and g > 255 and b < 30
```

is a convenient way to return a boolean expression without actually using `if` statements

(Learning Outcome!)

Different arguments

How about rewriting our function so that it takes as arguments

- the entire image
- x coordinate
- y coordinate

```
def is_green(img, x ,y):
    """
    Return true if the rgb value are within 30 of the color <0,255,0>
      in other words, the r value can be 0-30, the g value 225-255, and the b value 0-30
    input:
      img - the image
      x - x coordinate
      y - y coordinate
    return True or False
    """
    pixel = img[x][y]
    r = pixel[0]
    g = pixel[1]
    b = pixel[2]
    return r < 30 and g > 255 and b < 30
```

The "Best" Arguments to accept?

Question: Which is better?

```
def is_green(img, x ,y):
```

vs

```
def is_green(r,g,b):
```

Multiple assignment

Advanced tip: Python has a nice convenient ("ergonomic!") feature to put elements in a list (if you know it's length) into variables

called "tuple unpacking" or "multiple assignment"

Can save some lines in our above function. This is optional.

```
r, g, b = img[x][y]
```

Review questions (Slide 1)

1. Name 2 applications areas of computer vision
2. If given an image in variable named `awesome_image`, how do we get the pixel that's pixels down, 8 to the right?
3. What happens if we define a variable in a function and then try to use it in the global scope?
4. What happens if we define a variable in a for loop and then try to use it after the for loop?

Review questions (Slide 2)

5. Which colour is 0,0,50 closet to? (dark / light + red / green / blue)
6. In a blank Python file, initialize a list of lists of lists that gives a small, 2x2 4 pixel image with red in top left, green in top right, blue in bottom left, and black in bottom right.

Review questions (Slide 3)

7. Look at this list

```
img = [ [[10,20,30], [15,20,200]],
  [[25,0,98], [5,0,0]],
  [[60,200,0], [0,0,200]] ]
```

7a. How many pixels does it have? 7b. What's the height? 7c. What's the width? 7d. How many mostly blue pixels? 7e. How many pixels with no red?

Nested loops and images

We can use nested loops to initialize an image

```
for row in range(100):
    for col in range(100):
        img[row][col] = [0,0,0]
```

Traversing image with nested loops

We can also traverse our image this way.

Let's put our snippets together into a single file.

```
for row in range(100):
    for col in range(100):
        img[row][col] = [0,0,0]
```

Try to write a file!

Ok! We can give it a first go.

Any problems?

Two options for debugging:

1. Think really hard about it (sometimes a good idea; when you ask me questions I sometimes start with this approach. But if the code is long or there's a long line of people I might ask you to start using "print debugging")
2. Look closely at the output that doesn't meet expectations

Reorganizing our code into a module

- Put our function in a separate file
- Now we can `import xyz`
- Let’s call it "green_checker" – the one stop shop for all your is-my-pixel-green needs
- Something to think on: how could we extend this module to handle many colours?

- Short names: you can rename modules as you import them using `as`
  - `import pandas as pd`
  - `import mycoolcoloursmodule as colour`

So if you want to give your module a long name: super_awesome_green_checker, you might do

```
import super_awesome_green_checker as gc
```

then run

```
gc.is_green
```

What's in cmpt120image.py?

It's actually just a "wrapper" for `pygame`.

- pygame is a "package" – a collection of modules
- Anyone heard of a "GPT wrapper?"

More questions

- What is the name of a module relative to the filename?
- What can nested loops be used for in the context of image processing?
- What is the difference between a package and a module?

What does answer contain after the code is run below?

```
def special(numbers):
  for x in numbers:
    for y in numbers:
      if x < y:
        return x+y

answer = special([3,1,2])
```

a) How many bytes does the string "Aha!" need to be coded using the ASCII code? b) Write 32 using 1-byte binary.

What will be the result of executing this code? If there is some problem, how can it be fixed?

```
def function_a(p):
    x = p + 100
    return x
result = function_a(1)
print(result)
```

What will be the result of executing this code? If there is some problem, how can it be fixed?

```
def function_b(p):
      p = p + 100
      return p
arg = 1
print("arg before function",arg)
value = function_b(arg)
print(value)
print("arg after function",arg)
```

What will be the result of executing this code?

```
def functionC(p):
    x = p + 100
     print(x)
arg = 1
value = functionC(arg)
print(value)
```

What does the user see when this code is executed?

```
def function_e(lst):
   found = "no"
   for word in lst:
        if "a" in word:
            return("the  word " + word + " has an 'a'")
            found = "yes"
        if "b" in  word:
            return("the  word " + word + " has a 'b'")
            found = "yes"
   if found == "yes":
      result = "I found at least a word with an 'a' or at least a 'b'"
   else:
      result = "I did not find any word in the list with 'a' nor with 'b'"
   return result

words = ["perfect", "great", "absolutely"]
messg = function_e(words)
print(messg)
```

What does the user see when this code is executed?

```
def add_stars(word):
   result = word + "**"
   return result

lst = ["I", "am", "very", "happy"]
order = [2, 3, 0, 1]
message = ""
for i in range(len(order)):
     starred = addStars(lst[order[i]])
     message = message + starred

print(message)
```

Brief foray into memory

- On our hard drive, we have RAM
- RAM = Random Access Memory (fast)
- Each object in our program has an address in memory
  - Variables references / points to that address
  - we can use python `id` to get that address, e.g.

```
x = 5 # means there's a "bucket/slot" in our RAM somewhere that is holding the value 5
print(id(x)) # prints out that id
```

Here’s where things get a little weird with list

if we do this:

```
a = [1,2,3]
b = a
```

B is an alias of a. Points to the same address. How to check?

```
id(a) == id(b)
```

But if we do this:

```
a = [1,2,3]
c = a[:]
d = list(c)
```

Both c and d are copies of a. New address, same content.

Why might this matter?

An alias doesn’t take extra space (pro) but if we edit the original, we also edit the alias (sometimes good, sometimes bad)

A copy takes extra space, but we can edit them separately

To the Python code visualizer!

Some random examples of slight memory efficiencies we can achieve

- Use append over list concatentation

Lists are passed by reference

Important Python fact: lists are passed by references

Most variables are not!

What about lists of lists?

The list slicing and list() ways of cloning lists only works for 1-dimensional lists. For lists that contain sublists, only a shallow copy will take place of the outer list. The elements in the sublists will still point to the original objects.

In the final project, you will need to be aware of this.

If you do not wish to change the original image, create a new image using get_black_image(), for example.

Some 2 cents

- Sometimes efficiency is key (some CMPT classes, some coding interviews)
- But sometimes readability / convenience is key!
- Learning to balance is an art.

I use list concatenation sometimes when it’s not optimal in my personal research code…

But I might be a bit more thoughtful if I know my code will be run "in prod"

In general, this stuff isn't a common issue in Python – you can often not think about references and get away with it.

Some Python intros may not talk about it at all.

Most common place where this matters is when you're mutating multi-level data structures (list of lists, dicts of dicts)

When it matters, it can matter a lot – so knowing how to identify, test for, and fix alias / cloning related issues can make you stand out as a programmer!

In other CMPT courses, you'll use languages like C++ that are much more "explicit" about memory