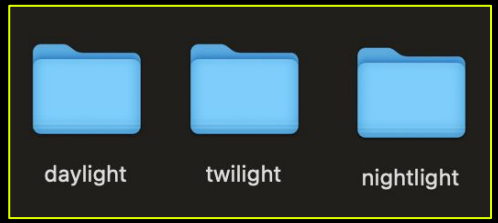
intro to code as a liberal art

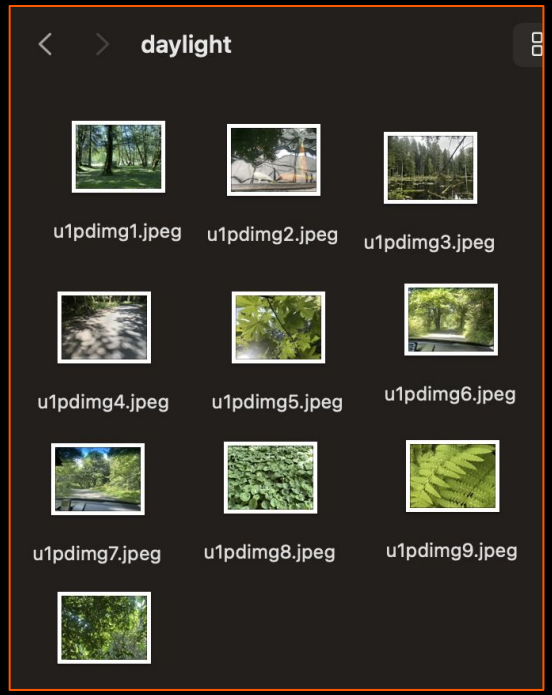
[unit 1 project]

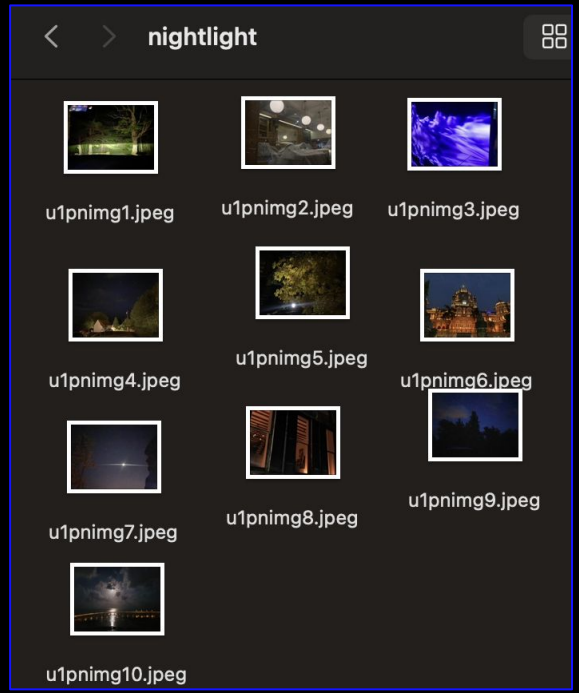
by.lara

idea.py

```
# My goal was just to find some cute pictures from my
gallery and improve my coding in general
```







algorithms (error handling)

```
10    # argument error message
11
12    if len(sys.argv) != 2:
13        exit("This command requires one argument : the name of your image folder.
14
```

1. argument error : input in the command line needs to be :
    a. python3
    b. the python algorithm file
    c. the desired image folder

```
[laraagrawal@Laras-MacBook-Air unit-1-project % python3 algo1.py
This command requires one argument : the name of your image folder. Image folder
 and algorithm must be in same folder.
[laraagrawal@Laras-MacBook-Air unit-1-project % python3 algo1.py daylight
laraagrawal@Laras-MacBook-Air unit-1-project %
```

algorithms (error handling)

```
# to avoid ds store file errors

def remove_ds_store(path):
    for files in os.walk(path):
        for file in files:
            if file == ".DS_Store":
                file_path = os.path.join(file)
                os.remove(file_path)
            else:
                exit

remove_ds_store(sys.argv[1])

try:

    files = listdir(sys.argv[1])
    files.remove(".DS_Store")

except:
    exit
```

2. ds_store errors :

a. happens when the algorithm randomly picks the ds_store file instead of an image file.

here i've used 2 different ways to delete the file, but i think the second way is doing the heavy lifting.

b. happens when the second method doesn't detect a ds_store file and returns a value error.

fixed using 'try-except' error handling, which essentially skips the step if the file doesn't exist.

algorithms (error handling)

unit-1-project > algo1.py > ...

```
40  # 'try-except' error handling to give personalized ValueError error message
41
42  try:
43      #  blend images as a first step
44
45      img = Image.blend(img1,img2,.5)
46
47
```

unit-1-project > algo2.py > ...

```
43
44  # 'try-except' error handling to give personalized ValueError error message
45
46  try:
47      # using blend as a test to see if images match in size (not part of edit)
48
49      sizetest1 = Image.blend(ogimg1,ogimg2,.5)
50      sizetest2 = Image.blend(ogimg1,img3,.5)
51      sizetest3 = Image.blend(ogimg2,img3,.5)
52
53
```

```
93  except:
94      exit("Images must be of the same size.")
95
```

3. value error :

happens when you try to blend images that don't match in size.

i used the 'try-except' error handling method to let the user know that the images need to match.

most of the main code goes in under 'try', and if an error occurs, it skips to 'except', which exits the error message.

although i don't use blend in algo2.py, i've included it there as a test for image matching.

algo1.maincode

```
4   import os
5   from os import listdir, path
6   import random
7   from PIL import Image
8   import sys
```

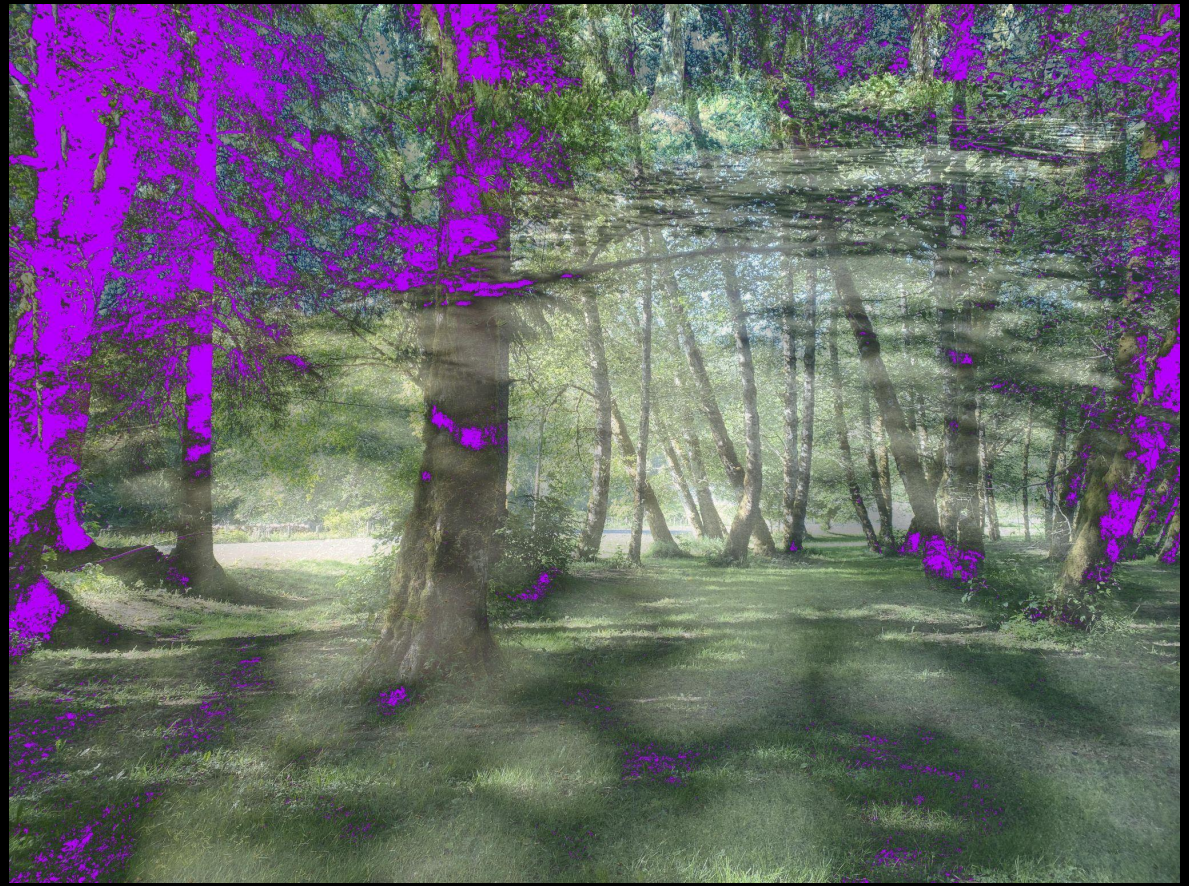
```
32  # opening 2 random files
33
34  random_file = random.choice(files)
35  img1 = Image.open (path.join(str(sys.argv[1]),random_file))
36
37  random_file = random.choice(files)
38  img2 = Image.open (path.join(str(sys.argv[1]),random_file))
39
```

algo1.py

a. import relevant modules/libraries
b. use random to open 2 files
c. blend them to create a base image
d. use path to save everything to same image folder
e. convert image to HSV
f. alter pixels using append and save new image as RGB
g. delete the base image

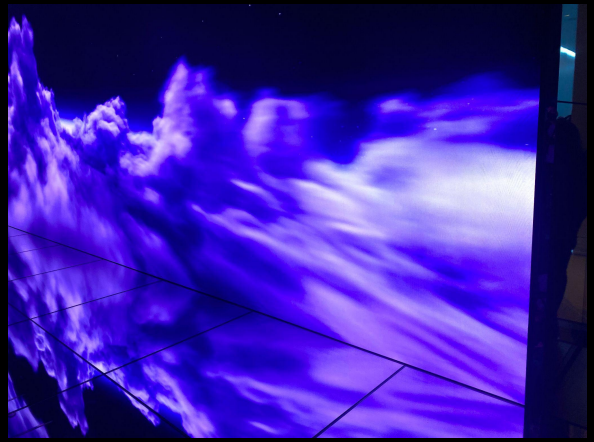
```
42  try:
43
44      #  blend images as a first step
45
46      img = Image.blend(img1,img2,.5)
47
48      # save output to same folder that the images were taken from
49
50      image_path = sys.argv[1]
51      img.save(f"{image_path}/tobedeleted.jpeg")
52
53      # alter pixels under certain value to fun colour and save
54
55      img_hsv = img.convert(mode="HSV")
56      img_hsv_data = img_hsv.getdata()
57      new_img_data = []
58
59      for p in img_hsv_data:
60          if p[2] < 50:
61              new_img_data.append((200, 255, 255))
62          else:
63              new_img_data.append(p)
64
65      img_hsv.putdata(new_img_data)
66      img_rgb = img_hsv.convert("RGB")
67      img_rgb.save(f"{image_path}/algo1-" + str(sys.argv[1]) + ".jpeg")
68
69      # delete the blended base image
70
71      def delete_image(image_path):
72          if os.path.exists(image_path):
73              os.getcwd()
74              os.remove(image_path)
75
76      image_path = sys.argv[1] + "/tobedeleted.jpeg"
77      delete_image(image_path)
```

algo1.daylight (input , output)



algo1.twilight (input , output)

algo1.nightlight (input , output)



algo2.maincode

```
4   import os
5   from os import listdir, path
6   import random
7   from PIL import Image
8   import sys
9   import PIL.ImageOps
```

```
33  # opening 3 random files
34
35  random_file = random.choice(files)
36  ogimg1 = Image.open (path.join(str(sys.argv[1]),random_file))
37
38  random_file = random.choice(files)
39  ogimg2 = Image.open (path.join(str(sys.argv[1]),random_file))
40
41  random_file = random.choice(files)
42  img3 = Image.open (path.join(str(sys.argv[1]),random_file))
```

algo2.py

a. import relevant modules/libraries
b. use random to open 3 files
c. Under 'try', use blend as an image size test
d. use path to save everything to same image folder

```
46  try:
47
48      # using blend as a test to see if images match in size (not part of edit)
49
50      sizetest1 = Image.blend(ogimg1,ogimg2,.5)
51      sizetest2 = Image.blend(ogimg1,img3,.5)
52      sizetest3 = Image.blend(ogimg2,img3,.5)
53
54      image_path = sys.argv[1]
```

algo2.maincode

```
56      # cropping images 1 & 2
57
58      width, height = ogimg1.size
59      crop = (0, 0, width, int(height/3))
60      cropimg1 = ogimg1.crop(crop)
61      cropimg1.save(f"{image_path}/cropimg1.jpeg")
62
63      width, height = ogimg2.size
64      crop = (0, int(height/3), width, int(height/3 + height/3))
65      cropimg2 = ogimg2.crop(crop)
66
67      # inverting cropped image 2
68
69      invertimg = PIL.ImageOps.invert(cropimg2)
70      invertimg.save(f"{image_path}/cropimg2.jpeg")
71
72      # pasting on image 3 and saving
73
74      width, height = img3.size
75      img1 = Image.open (f"{image_path}/cropimg1.jpeg")
76      img2 = Image.open (f"{image_path}/cropimg2.jpeg")
77      img3.paste(img1, (0,0))
78      img3.paste(img2, (0, int(height/3)))
79      img3.save(f"{image_path}/algo2-" + str(sys.argv[1]) + ".jpeg")
```

```
81      # deleting cropped images
82
83      def delete_image(image_path):
84          if os.path.exists(image_path):
85              os.getcwd()
86              os.remove(image_path)
87
88      delete1 = sys.argv[1] + "/cropimg1.jpeg"
89      delete2 = sys.argv[1] + "/cropimg2.jpeg"
90      delete_image(delete1)
91      delete_image(delete2)
92
```

algo2.py

e. use path to save everything to same image folder

f. crop images 1 & 2

g. invert image 2 colours

h. paste on image 3 & save

i. delete the cropped images

algo2.daylight (input , output)

algo2.twilight (input , output)

algo2.nightlight (input , output)