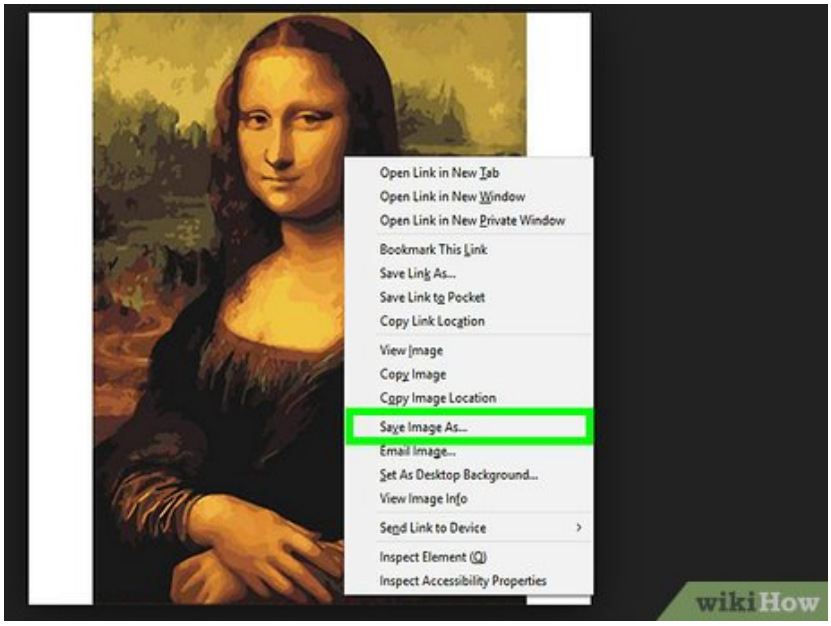
How to Make a Self?Writing Mosaic Program in Python

This article will show you how to use the graphics Python package 'Pygame' and basic file I/O to create a self-writing program that draws a mosaic of a given image. Python is a very versatile programming language with a great deal of...

Part 1 of 4:

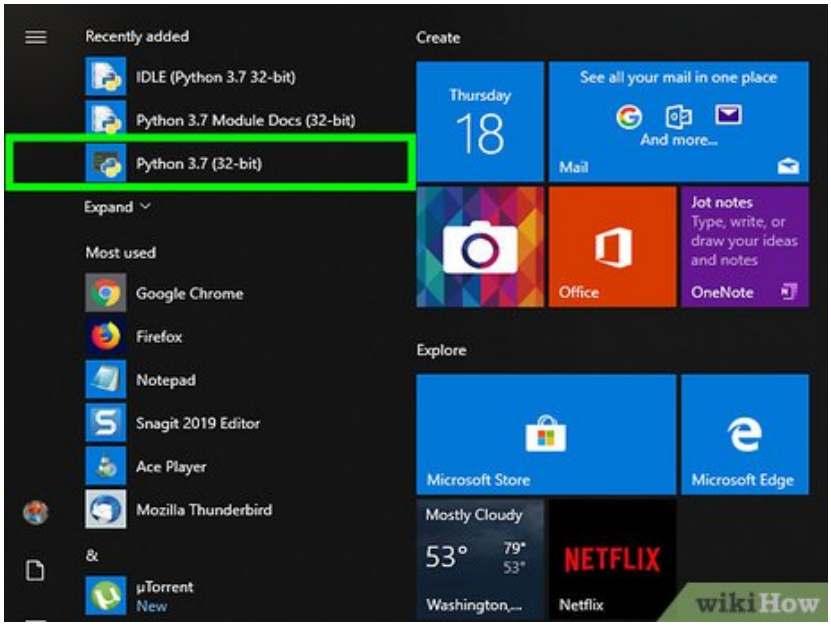
Downloading the Photo

1. 

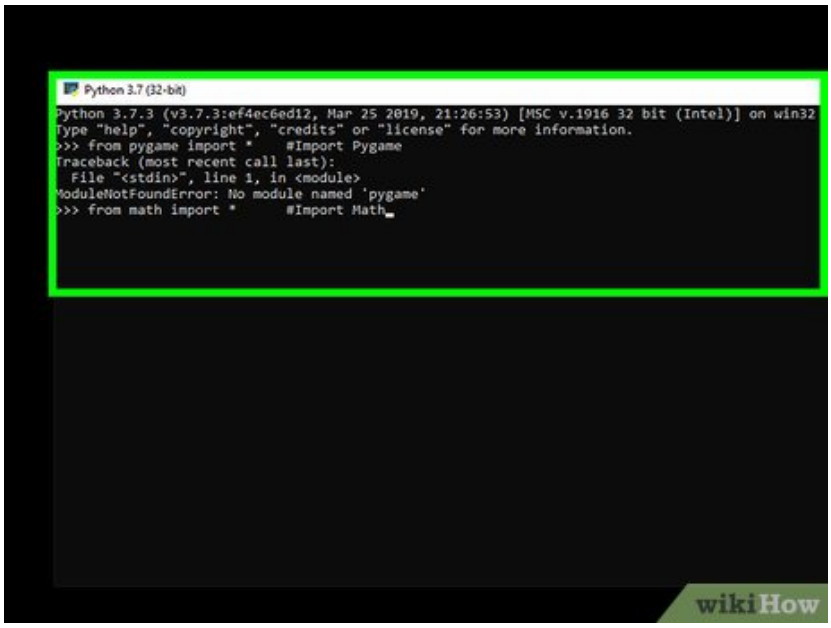
   Download the Mona Lisa Photo. This is the base picture you will be using to create your mosaic. Note the picture's dimensions in pixels as this will be important for your code later.
   1. Save it as "mona.jpg"
   2. The image is **743px by 1155px.** Ensure sure the it saves as this size or else the mosaic will not be drawn properly.

Part 2 of 4:

Creating the Main Program

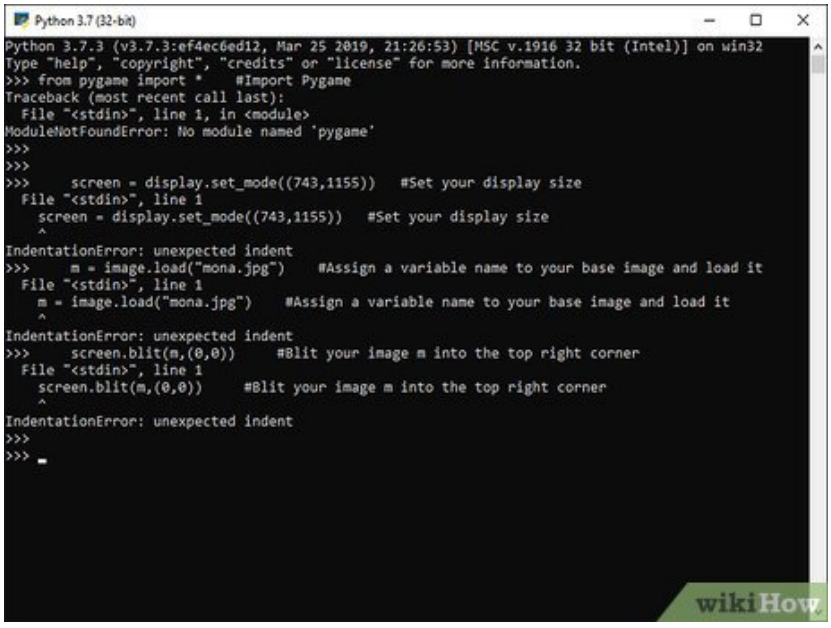
1. 

Open a new Python shell. Name the file "makeMona.py"; this file name is for your own reference.

2. 

Import your modules. For this program, you'll need the Pygame module to display your image and the Math module in order to perform your adding functionality.

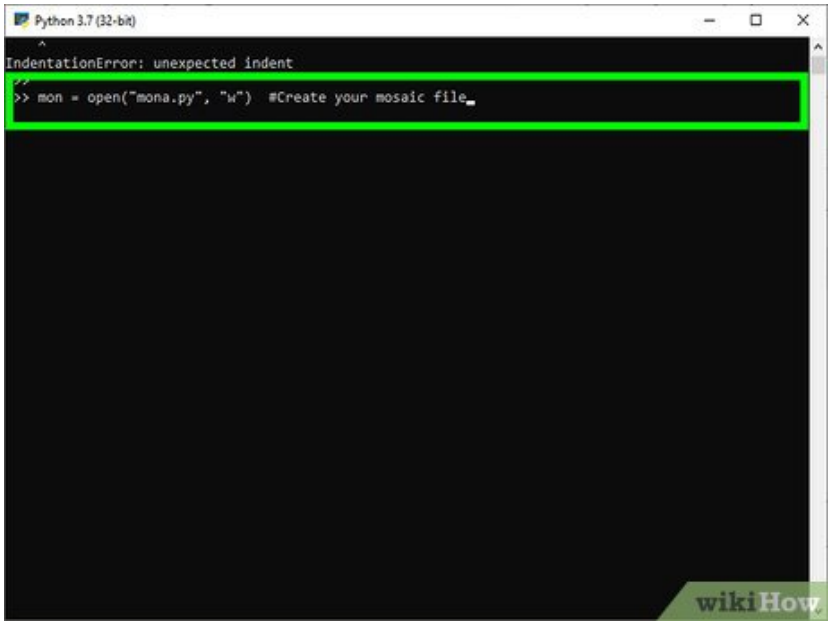
```
from pygame import * #Import Pygame from math import * #Import Math
```

3. 

Set up display and image. Before you can start mapping the image, you must create the display screen that the image will appear on and load in Mona Lisa.

1. `display.set_mode((743,1155))` is how you set your display. (743,1155) represents the screen size; note that it is the exact same size as your Mona Lisa image in pixels.

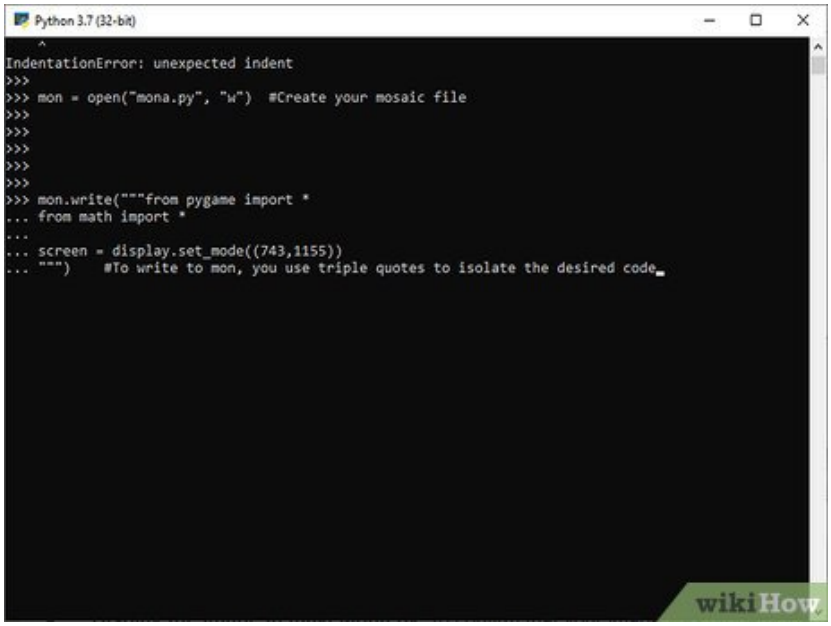
```
screen = display.set_mode((743,1155)) #Set your display size m = image.
load("mona.jpg") #Assign a variable name to your base image and load it
 screen.blit(m,(0,0)) #Blit your image m into the top right corner
```

4. 

Create a file for the self-writing program. Within your makeMona program, you will set up the **new file you will be writing to.**

1. Variable `mon` is your reference to the mosaic file. Within the quotations, you declare the file name "mona.py". The "w" declares that you will be writing to the new file.

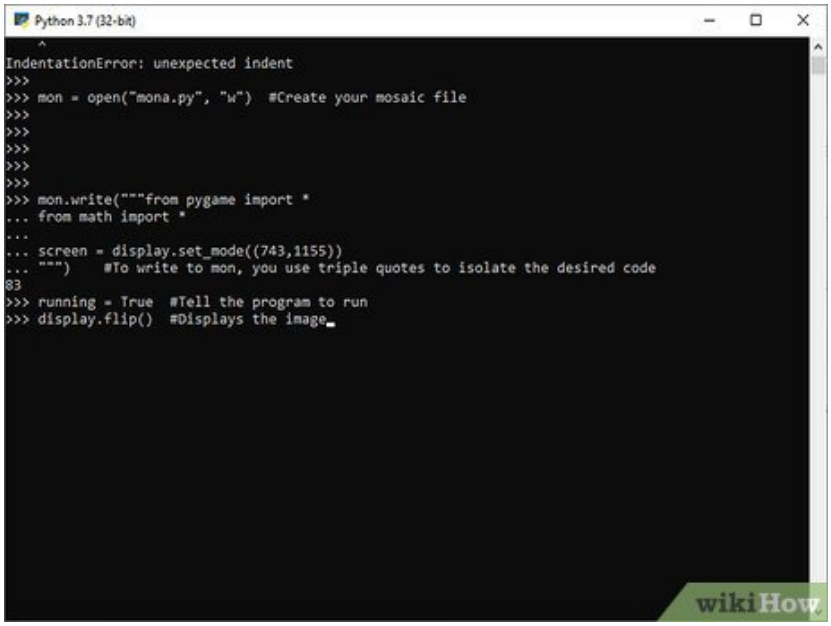
```
mon = open("mona.py", "w") #Create your mosaic file
```

5. 

Initialize the program. You can now start writing to mona.py. Here you will import your modules and set your display size.

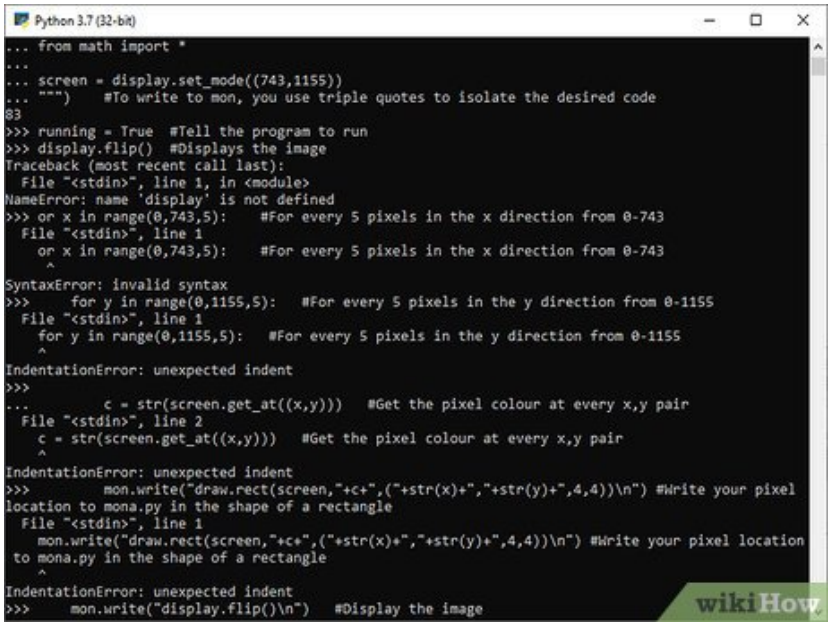
1. `mon.write` signifies that you are now writing the following code to your new file. Your display size will be the same as the Mona Lisa image.

```
mon.write("""from pygame import *  from math import *
screen = display.set_mode((743,1155)) """)
#To write to mon, you use triple quotes to isolate the desired code
```

6. 

Display the image in makeMona.py. To confirm that your image was properly loaded into your program, you will display the image onto the screen.

```
running = True #Tell the program to run display.flip()
#Displays the image
```

7. 

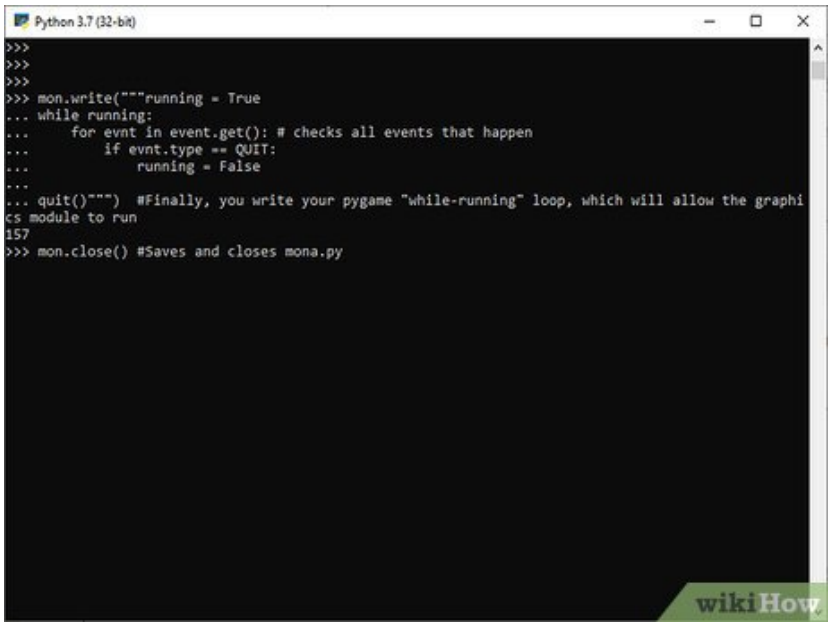
Map the mosaic pattern. You can now start parsing the image into small rectangles to create the mosaic. To get the best mosaic effect without distorting the image, you will map every 5 pixels into a rectangle.

1. `mon.write("draw.rect(screen,"+c+",("+str(x)+","+str(y)+",4` is the most important line to note. The breakdown is as follows:
2. `draw.rect(screen,"+c+",` signifies that you will draw your mosaic piece (individual rectangles to your screen, then appending "c" relates to `str(screen.get_at((x,y)))`, which is how you map the colour at each pixel

location.

3. The "x" and "y" in `("+str(x)+","+str(y)` represent the location of your rectangle. You use `str()` because you need the program to read these x and y values as strings (because you are inside your triple quotes, which is all string formatting).
4. The 4,4 in `+",4,4))n")` represent the dimensions of each rectangle in your mosaic. "n" is your newline character, which tells the program to go to the next line and start writing the next line of code.

```
for x in range(0,743,5):
#For every 5 pixels in the x direction from 0-743 for y in range(0,1155
,5): #For every 5 pixels in the y direction from 0-1155 c = str(screen.
get_at((x,y))) #Get the pixel colour at every x,y pair mon.write(
"draw.rect(screen,"+c+",("+str(x)+","+str(y)+",4,4))n")
#Write your pixel location to mona.py in the shape of a rectangle mon.
write("display.flip()n") #Display the image
```

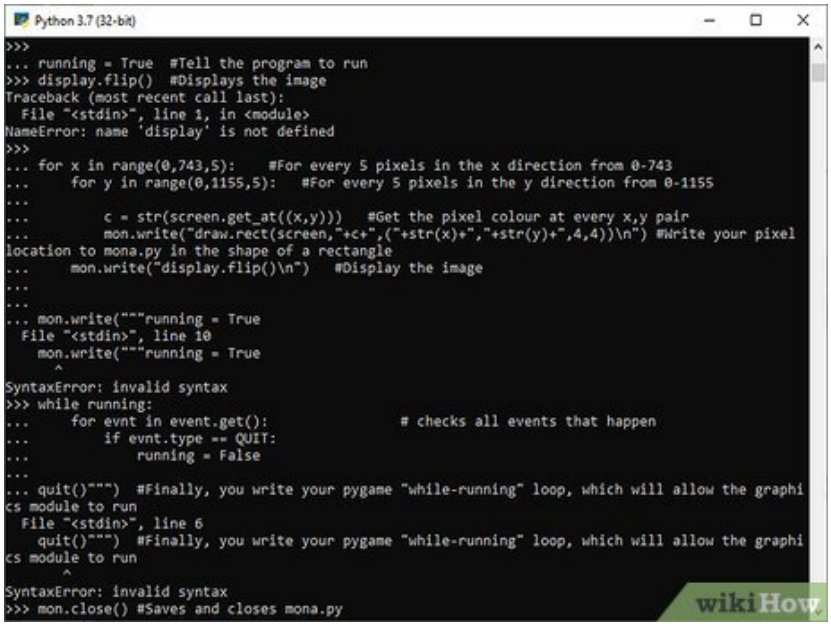
8. 

Create the while-running loop. As with any Pygame program, you must include your "while running" loop in mona.py.

```
mon.write("""running = True while running:
 for evnt in event.get(): # checks all events that happen
 if evnt.type == QUIT:  running = False quit()""")
#Finally, you write your pygame "while-running" loop, which will allow the g
 mon.close() #Saves and closes mona.py
```

Part 3 of 4:

Code Review and Testing

1. 

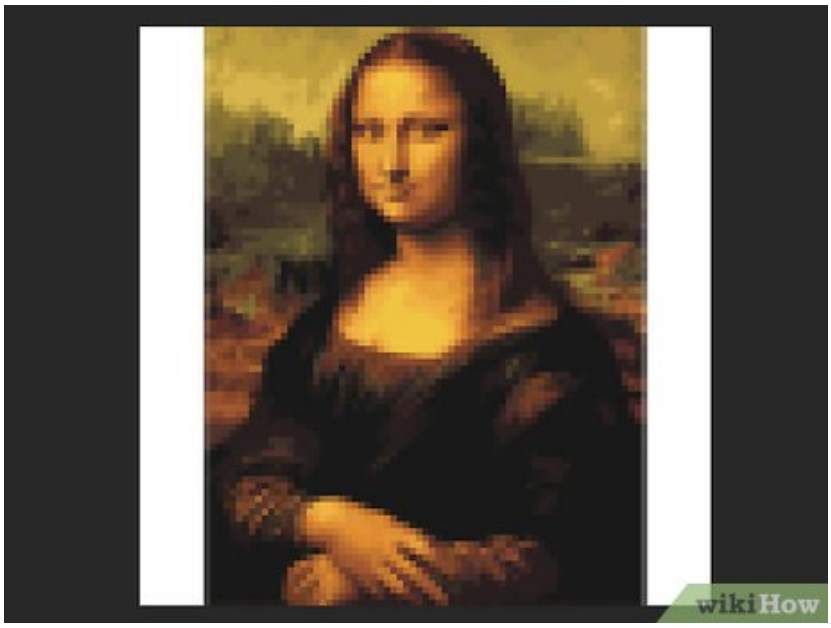
Review your code. Your main program makeMona.py is now complete. Here is all the code together.

```
from pygame import * #Import Pygame from math import * #Import Math
screen = display.set_mode((743,1155)) #Set your display size m = image.
load("mona.jpg") #Assign a variable name to your base image and load it
 screen.blit(m,(0,0)) #Blit your image m onto the top right corner mon
= open("mona.py", "w") #Createy your Mosaic file mon.write(
"""from pygame import *  from math import *
screen = display.set_mode((743,1155)) """)
#To write to mon, use triple quotes to isolate the desired code running
 = True #Tell the program to run display.flip() #Displays the image for
 x in range(0,743,5): #For every 5 pixels in the x direction from 0-743
 for y in range(0,1155,5):
#For every 5 pixels in the y direction from 0-1155 c = str(screen.
get_at((x,y))) #Get the pixel colour at every x,y pair mon.write(
"draw.rect(screen,"+c+",("+str(x)+","+str(y)+",4,4))n")
#Write your pixel location to mona.py in the shape of a rectangle mon.
write("display.flip()n") #Display the image mon.write("""running = True
 while running:
 for evnt in event.get(): # checks all events that happen
 if evnt.type == QUIT:  running = False quit()""")
#Finally, you write your pygame "while-running" loop, which will allow the g
 mon.close() #Saves and closes mona.py
```

2. 

Open the new mona.py program. If you go to wherever your makeMona program is saved, directly below it you will find your new program file mona.py.

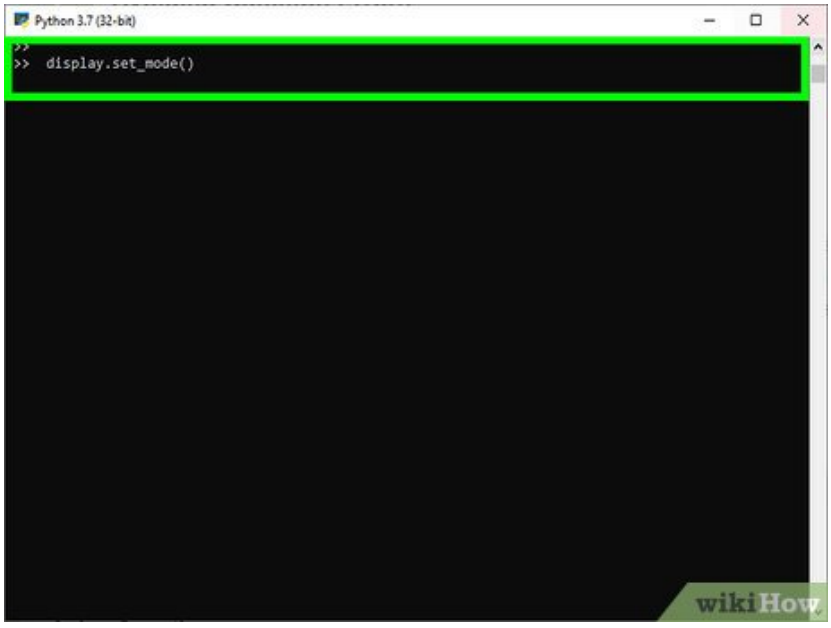
1. If you open this file, you will see hundreds of lines of code. Magic! This code is each individual piece of the mosaic. Each line represents a new rectangular piece of Mona Lisa.
2. It may take several seconds to open; this is normal because it is such a big file.

3. 

Run mona.py. If everything up to this point is looking good, your last step is to run your program and behold your new Mona Lisa mosaic.

Part 4 of 4:

Avoiding Common Errors

1. 

Troubleshoot your program. It's easy to make small little mistakes once you're implementing the code for your own mosaic. Here are some common issues people run into and how to resolve them.

1. **Where is my new program?** - After running the main program, the new one should appear in the same file location as the main program.
2. **My new program isn't running** - All code you wrote for the self-writing program is contained in triple quotes. Review all of the code inside of these quotes throughout the program and ensure you didn't miss things like colons, closing brackets/quotes, or indentation formatting.
3. **Why is the image distorted?** - This can be a result of your display size. Ensure that your display size in `display.set_mode()` all throughout your code is the same.