!!!!!!!!!!!! Every image must be in a project directory or you must change ‘curdir + imgdir’ by hand in the code!!!!!!!!!!!!!!!!!

1 . I wrote custom matrix multiplication and matrix-vector multiplication by hand because I am not sure if we are allowed to use np.dot but np.dot is faster in some cases

2. I wrote some iterative methods, only richardson’s is in a matrix form others are element-wise

3. I create my own nxn K matrix, I choose this k matrix to be diagonally dominant because we use iterative methods to decode an encoded code... -> in diagonals I write 3 \* n and I fill rest of the matrix by random integers from 0 to 5 which in avrg sould give 2.5 for each non-diagonal element, with this method it is very unlikely to get non-diagonally dominant matrix especially when n is a big number

And I check if k is invertible with gram\_schmidt\_basis function which returns None if there are dependant vectors in gram schmidt basis

4.encoding -

Imagine we have to encode 200 integers, we would have to create 200x200 matrix, which is not optimal, neither for memory management nor for the speed of the code, so I give an opportunity for you to choose your own n, then those 200 integers would be chopped into n-size pieces and I encode them one by one and then merge them together, but 200%n might not always be 0, so when the last piece of those 200 integers roles in I create new k for whatever length is left to encode, and I encode it with that, both k-s are stored in an array for encoding in global variable encodingks

For matrix - I just flatten entire matrix and store its height and width in global variables for rebuilding ,

I also had some ideas for rebuilding without global variables like changing last bit of an image that we are hiding this matrix into to length and width like mat[len(mat)][len(mat[0])] = (r , heigth , width)

But height and width might be more than 256 and overflow would be noticable so we would have to do the same hiding method for length and width but we would start from the last bit , but theres a problem that they might meet eachother and it was not nessasery so I did not waste my time coding this

For string -

I store string length into global variable for decoding afterwards

5. decoding-

Decoding is the same by reverse of encoding, I check the length of encodingks[0] to know how I must chop up my encoded array and I decode it one by one until we get to the last iteration in where if our (encoded array length !% encodingks[0]) I know second k is in encodingks[1] so I decode using that

6. hiding img -

Lets say our host is a - horse and what wer trying to hide is encoded array – arr

I take elements of the array and turn it into 18length binary bits, (after encoding we might get absurdly lasrger numbers, I had a problem first time I wrote this because I was doing it on 12 length binary numbers). Then I split this 18 bit string into 3, 6-6-6 I iterate over our horse and write each 6 into one rgb element but, again splitting 6 into 2-s and changing each r,g,b-s binary representations last two bits into corresponding strings of 2, so for each encoded element we need 3img pixels to store it

7. getting img back-

I wrote two function one for string and other for matrix, but they differ in only one line,

Both of the return encoded array but for one we would need to iterate over the image strilength-s time (this is the global variable that I created for string length) and for other matheight \* matwidth \*3 times

Other than that its just reverse of hiding image, I take 3pixels and I strip each rgb-s binary representations last two bits and I concat them to get a single element of encoded array, then I decode it and if it’s a string I just do chr() for each element and we would get back the string and if it’s a matrix I have helper functions like build() to rebuild the image

-\_\_\_\_\_\_- NOTICE

After canceling the run we would lose our global variables, so if we want to get back encoded arrays after termination we would have to store every global variable that I created somewhere, I already wrote that this could be done in the image, the same way we hide the encoded array, but there is anotehr way, we just write them in the textfile, which again was not necessary and I did not waste time writing it.