**CSE 8A: Intro to Programming I** **Lecture Note # 23**
***Paul Cao, Fall 2021*** **Friday 11/19/2021**

# Image Encryption and Decryption - 2

## Hiding information in images

The idea is to hide some information inside each pixel and use the shortcomings of human observation to make it *invisible*

| Red | Green | Blue | |
|---|---|---|---|
| (39, | 56, | 101) | 🟦 |
| (37, | 59, | 100) | 🟦 |

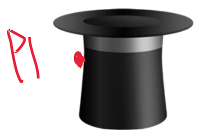
If we use 2 bits per color channel, what color should (11, 00, 01) be in theory?
[✓ A. Pink(ish)] B. Blue C. Green D. White E. Black

lots of red no green Some blue

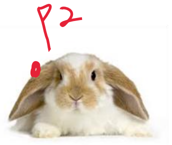
If we use 2 bits per color channel, what color should (11, 00, 01) be in Python?
A. Pink(ish) B. Blue C. Green D. White [✓ E. Black]

(00000011, 00000000, 00000001)

| | Red | Green | Blue | |
|---|---|---|---|---|
| Pixel in context image P1 | ( 00100111, | 00111000 | , 01100101 ) | (3, 0, 1) |
| Pixel in secret message P2<br>(at same x, y position) | ( 01100100, | 11111001 | , 00001111 ) | |



Context image



Secret message

## Encryption

1. Obtain the most important bits of the secret message

(011, 111, 000)

2. Put these important bits into the least important bits of the context image

P1 (00100 011, 00111 111, 01100 000)

## Decryption

1. Obtain the least significant bits of the context image

(011, 111, 000)

2. Shift them to make a pixel in the reduced color picture

(011 00000, 111 00000, 000 00000) new bunny

**Exercise:** Use the two bit encoding scheme to encode (7, 12, 15) into (65, 129, 252).

00000111, 00001100, 00001111 01000001 10000001 11111100
00 00 00

What is the encrypted pixel?

64 128 252

(01000000, 10000000, 11111100) (00, 00, 00)

What is the decoded pixel?

(00, 00, 00)

(00000000, 00000000, 00000000) (0, 0, 0)

**Bitwise operations in Python**

**1. bit shifting**

You can shift a number to the left or right using the shift operator

<< left shift : remove bits from left, append 0 on the right : * 2

>> right shift : opposite of left : / 2

What is `6 << 2`?

A. 12 B. 12 C. 64 [✓ D. 24] E. None of the above

00 0 11000
16+8

What is `20 >> 1`

A. 20 B. 100 [✓ C. 10] D. 2 E. None of the above

00001010

What is `10>>2`?

A. 5 B. 2.5 [✓ C. 2] D. 1 E. None of the above

5/2

What is `10 << 10`?

A. 0 [✓ B. 10240] C. 255 D. Something else

in theory ~~0000~~ ~~1010~~

x=01010001
(x<<6)>>6
won't work

(25<<6)>>6

What will this print (all numbers represented in decimal)?

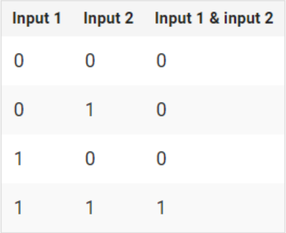
```
x = 25
x >> 2
print(x)
```

X=

A. 6 [✓ B. 25] C. 3 D. 2 E. Something else

## 2. Bitwise operation

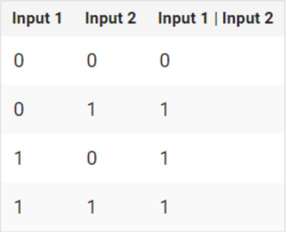
&: bitwise and



| Input 1 | Input 2 | Input 1 & input 2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

0 : false
1 : true

| bitwise OR



| Input 1 | Input 2 | Input 1 \| Input 2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

What is 0b11001100 & 0b11110000

A. 0b00000000
B. 0b11110000
C. 0b11001100
[✓ D.] 0b11000000
E. None of the above

11001100
& 11110000
11000000

11001001
& 00000011
00000001

What is 0b11001100 | 0b11110000

A. 0b11110000
B. 0b11111100
C. 0b11000000
D. 0b00111100
E. None of the above

11001100
| 11110000

**Exercise**

Take the last two bits from red and return it

```
def getLast2(red):
```

**Exercise**

Put the first two bits of red1 into the last two bits of red2

```
def put2Digits(red1, red2):
```

## Steganography summary

context img

secret img

**Encryption process**

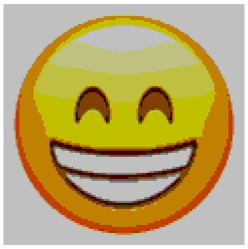
For every pixel in secret and context (same locations):

1. grab the leading two positions of secret for r, g, b
2. put the result from step 1 into the trailing two positions of context's r, g, b

return the encrypted image

encrypted img

decrypted img



**Decryption process**

create a blank canvas with the same size as the encrypted image

For every pixel in the encrypted image

1. grab the trailing two positions of r, g, b
2. shift the result from step 1 to the left 6 positions
3. put the result from step 2 into the corresponding pixel in the blank canvas

return the decrypted image (i.e. the modified blank canvas)