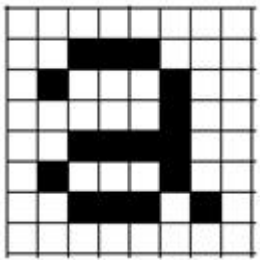
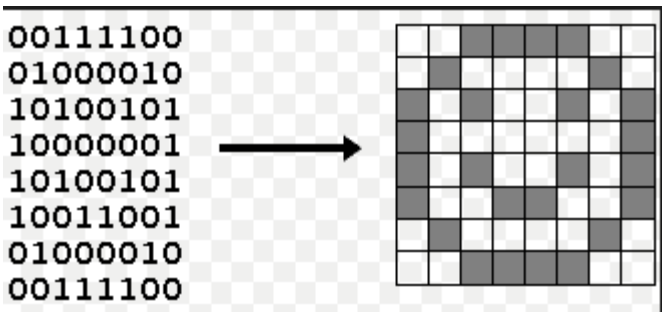
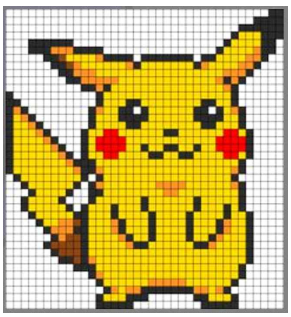
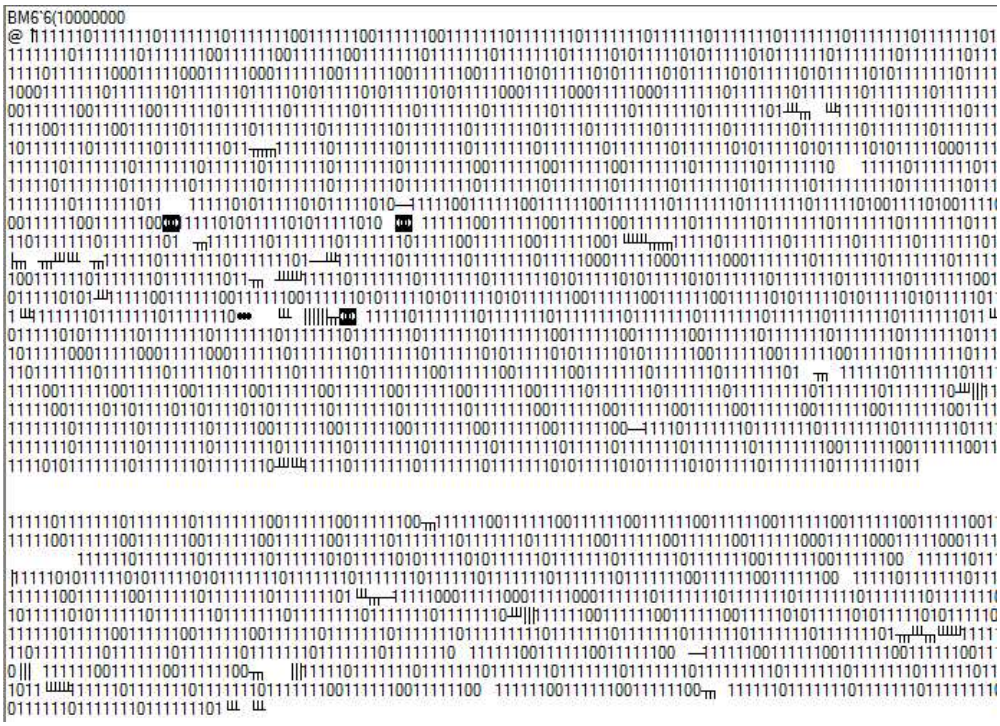
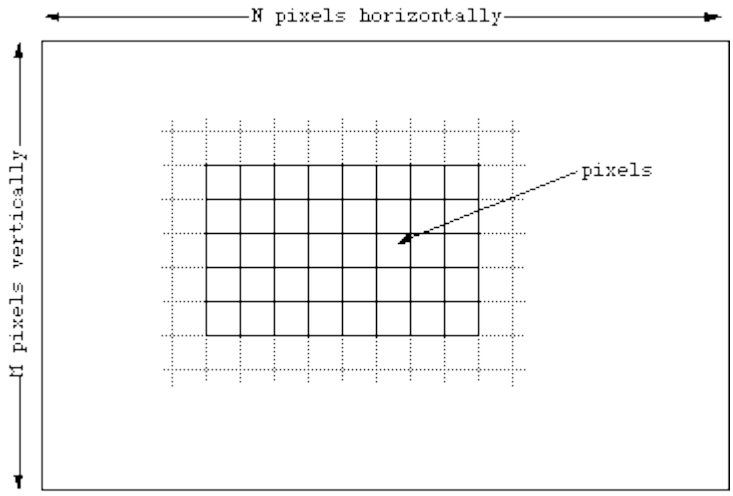



00111100
01000010
10100101
10000001
10100101
10011001
01000010
00111100

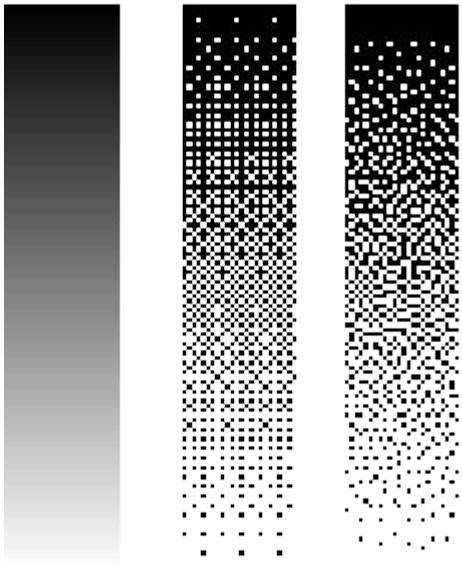



BM6`6(10000000

해상도

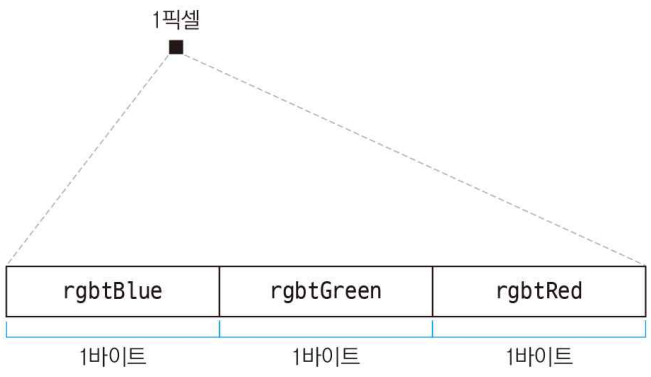

N pixels horizontally
M pixels vertically
pixels

BitMap

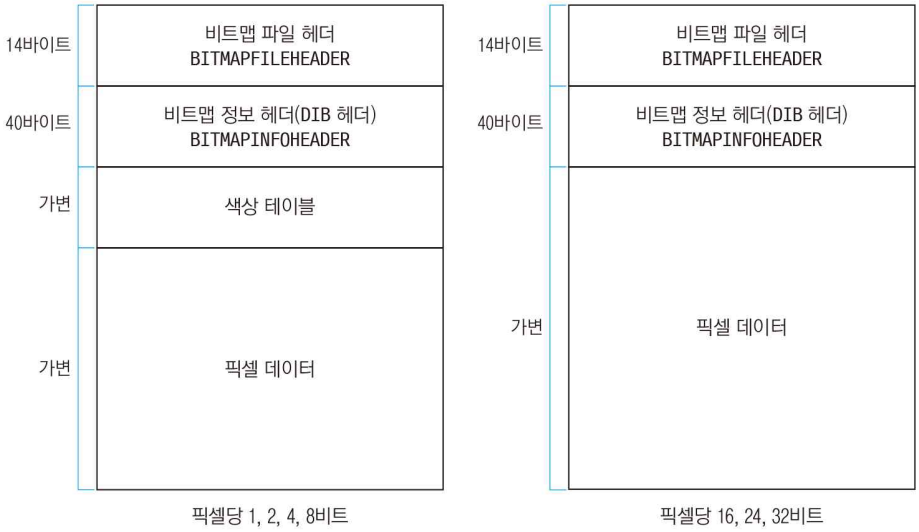


GrayScale

24bit Color



BMP 파일 구조



비트맵 파일 헤더의 구조

멤버	크기(바이트)	설명
bfType	2	BMP 파일 매직 넘버. 비트맵 파일이 맞는지 확인하는데 사용하며 ASCII 코드로 0x42(B), 0x4D(M)가 저장됩니다.
bfSize	4	파일 크기(바이트)
bfReserved1	2	현재는 사용하지 않으며 미래를 위해 예약된 공간
bfReserved2	2	현재는 사용하지 않으며 미래를 위해 예약된 공간
bfOffBits	4	비트맵 데이터의 시작 위치

비트맵 정보 헤더의 구조

멤버	크기(바이트)	설명
biSize	4	현재 비트맵 정보 헤더(BITMAPINFOHEADER)의 크기
biWidth	4	비트맵 이미지의 가로 크기(픽셀)
biHeight	4	비트맵 이미지의 세로 크기(픽셀). 양수: 이미지의 상하가 뒤집혀서 저장된 상태 음수: 이미지가 그대로 저장된 상태 보통 세로 크기는 양수로 저장되어 있습니다.
biPlanes	2	사용하는 색상판의 수. 항상 1입니다.
biBitCount	2	픽셀 하나를 표현하는 비트 수
biCompression	4	압축 방식. 보통 비트맵은 압축을 하지 않으므로 0입니다.
biSizeImage	4	비트맵 이미지의 픽셀 데이터 크기(압축 되지 않은 크기)
biXPelsPerMeter	4	그림의 가로 해상도(미터당 픽셀)
biYPelsPerMeter	4	그림의 세로 해상도(미터당 픽셀)
biClrUsed	4	색상 테이블에서 실제 사용되는 색상 수
biClrImportant	4	비트맵을 표현하기 위해 필요한 색상 인덱스 수

struct 모듈

C언어의 struct를 Python에서 사용하기 위한 byte 단위 표현. 이 struct로 파일이나 네트워크 연결에 사용하는 이진 데이터를 다룰 수 있음

struct.pack(format, v1, v2, ...)
struct.unpack(format, buffer)

문자	바이트 순서	크기	정렬
@	네이티브	네이티브	네이티브
=	네이티브	표준	none
<	리틀 엔디안	표준	none
>	빅 엔디안	표준	none
!	네트워크 (= 빅 엔디안)	표준	none

포맷 문자

포맷	C형	파이썬 형	표준 크기	노트
x	패드 바이트	값이 없습니다		
c	char	길이가 1인 bytes	1	
b	signed char	정수	1	(1), (2)
B	unsigned char	정수	1	(2)
?	_Bool	bool	1	(1)
h	short	정수	2	(2)
H	unsigned short	정수	2	(2)
i	int	정수	4	(2)
I	unsigned int	정수	4	(2)
l	long	정수	4	(2)
L	unsigned long	정수	4	(2)
q	long long	정수	8	(2)
Q	unsigned long long	정수	8	(2)
n	ssize_t	정수		(3)
N	size_t	정수		(3)
e	(6)	float	2	(4)
f	float	float	4	(4)
d	double	float	8	(4)
s	char[]	bytes		
p	char[]	bytes		
P	void *	정수		(5)

원리에 충실한 구현

```
import turtle
import struct

inFp = open("images/lenna_64_Grey.bmp", 'rb')

bmpHeader = inFp.read(14)
bmpInfoHeader = inFp.read(40)

biOffBits = struct.unpack('i', bmpHeader[10:14])[0]
biWidth = struct.unpack('i', bmpInfoHeader[4:8])[0]
biHeight = struct.unpack('i', bmpInfoHeader[8:12])[0]
biBitCount = struct.unpack('H', bmpInfoHeader[14:16])[0]
biSizeImage = struct.unpack('i', bmpInfoHeader[20:24])[0]

inFp.seek(biOffBits)
bmpData = inFp.read(biSizeImage)

print("이진 데이터 시작 위치 : %d" % biOffBits)
print("가로 해상도 : %d" % biWidth)
print("세로 해상도 : %d" % biHeight)
print("픽셀당 bit 크기 : %d" % biBitCount)
print("이미지 데이터 크기 : %d" % biSizeImage)
print(bmpData)

turtle.speed(0)
for y in range(0, biHeight):
    for x in range(0, biWidth) :
        dotColor = struct.unpack('<B', bmpData[y * biWidth + x: y * biWidth + x + 1])[0]
        turtle.goto(x, y)
        turtle.pendown()
        turtle.dot(2,"#%02x%02x%02x" % (dotColor, dotColor, dotColor))
        turtle.penup()
    pass

turtle.done()
```

약간의 성능 개선

```
import turtle
import struct

inFp = open("images/lenna_64_Grey.bmp", 'rb')

bmpHeader = inFp.read(14)
bmpInfoHeader = inFp.read(40)

biOffBits = struct.unpack('i', bmpHeader[10:14])[0]
biWidth = struct.unpack('i', bmpInfoHeader[4:8])[0]
biHeight = struct.unpack('i', bmpInfoHeader[8:12])[0]
biBitCount = struct.unpack('H', bmpInfoHeader[14:16])[0]
biSizeImage = struct.unpack('i', bmpInfoHeader[20:24])[0]

inFp.seek(biOffBits)
bmpData = inFp.read(biSizeImage)

print("이진 데이터 시작 위치 : %d" % biOffBits)
print("가로 해상도 : %d" % biWidth)
print("세로 해상도 : %d" % biHeight)
print("픽셀당 bit 크기 : %d" % biBitCount)
print("이미지 데이터 크기 : %d" % biSizeImage)
print(bmpData)

turtle.speed(0)
for y in range(0, biHeight):
    turtle.goto(0, y)
    turtle.pendown()
    for x in range(0, biWidth) :
        dotColor = struct.unpack('<B', bmpData[y * biWidth + x: y * biWidth + x + 1])[0]
        turtle.goto(x, y)
        turtle.pencolor("#%02x%02x%02x" % (dotColor, dotColor, dotColor))
    turtle.penup()
    pass

turtle.done()
```

Tk 모듈 사용

```
import struct
import tkinter

inFp = open("images/lenna_256_Color.bmp", 'rb')

bmpHeader = inFp.read(14)
bmpInfoHeader = inFp.read(40)

biOffBits = struct.unpack('i', bmpHeader[10:14])[0]
biWidth = struct.unpack('i', bmpInfoHeader[4:8])[0]
biHeight = struct.unpack('i', bmpInfoHeader[8:12])[0]
biBitCount = struct.unpack('H', bmpInfoHeader[14:16])[0]
biSizeImage = struct.unpack('i', bmpInfoHeader[20:24])[0]
inFp.seek(biOffBits)
bmpData = inFp.read()

print("이진 데이터 시작 위치 : %d" % biOffBits)
print("가로 해상도 : %d" % biWidth)
print("세로 해상도 : %d" % biHeight)
print("픽셀당 bit 크기 : %d" % biBitCount)
print("이미지 데이터 크기 : %d" % biSizeImage)

window=tkinter.Tk()
window.title("Lenna")
window.geometry("%dx%d"%(biWidth, biHeight))

canvas=tkinter.Canvas(window, relief="solid", bd=2)

if biBitCount == 8 :
    for y in range(0, biHeight):
        for x in range(0, biWidth):
            dotColor = struct.unpack('<B', bmpData[y * biWidth + x: y * biWidth + x + 1])[0]
            canvas.create_line(x, y, x + 1, y, fill="#%02x%02x%02x" % (dotColor, dotColor,
dotColor))
elif biBitCount == 24:
    for y in range(0, biHeight):
        for x in range(0, biWidth):
            dotColor = struct.unpack('<BBB', bmpData[y * biWidth * 3 + x * 3: y * biWidth * 3
+ x * 3 + 3])
            canvas.create_line(x, y, x + 1, y, fill="#%02x%02x%02x" % (dotColor[2], dotColor[1],
dotColor[0]))
else:
    pass

canvas.pack()

window.mainloop()
```

openCV 모듈 사용

opencv-python 패키지 설치
opencv-contrib-python 패키지 설치

```
import cv2

original = cv2.imread('images/lenna_256_Color.bmp', cv2.IMREAD_COLOR)

cv2.imshow('Lenna', original)

cv2.waitKey(0)
cv2.destroyAllWindows()
```