Using Text to "Map" Between any Media

- We can map anything to text.
- We can map text back to anything.
- This allows us to do all kinds of transformations:
  - Sounds into .csv files and back again
  - Sounds into pictures.
  - Pictures and sounds into lists (formatted text), and back again.

Why Care About Media Transformations?

- It's another form of encoding and decoding (same as we've been doing all semester)
- Transformed digital media can be more easily transmitted
  - For example, transfer of binary files over email is often accomplished by converting to text and then back to binary later
- We can encode additional information to check for and even correct errors in transmission.
- It may allow us to use the media in new contexts, like storing it in databases.
- Some transformations of media are made easier when the media are in new formats.

Mapping Sound to Text

- Sound is simply a series of numbers (sample values).
  - For 16-bit sounds: between +32,767 and -32,768
- To convert them to text means to simply create a long series of numbers.
- We can store them to a file to manipulate them elsewhere.

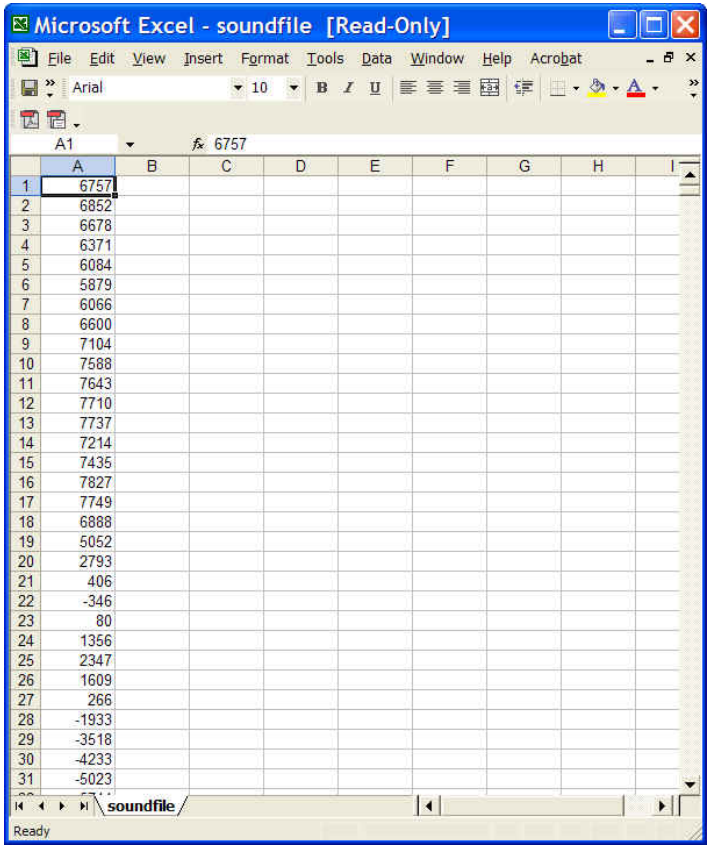
Copying a Sound to Text

```
def soundToText(sound,filename):
  file = open(filename,"wt")
  for s in getSamples(sound):
    file.write(str(getSample(s))+"\n")
  file.close()
```

DMS 102: Practice reading and writing files using Python was done in an earlier section

What to do with Sound as Text

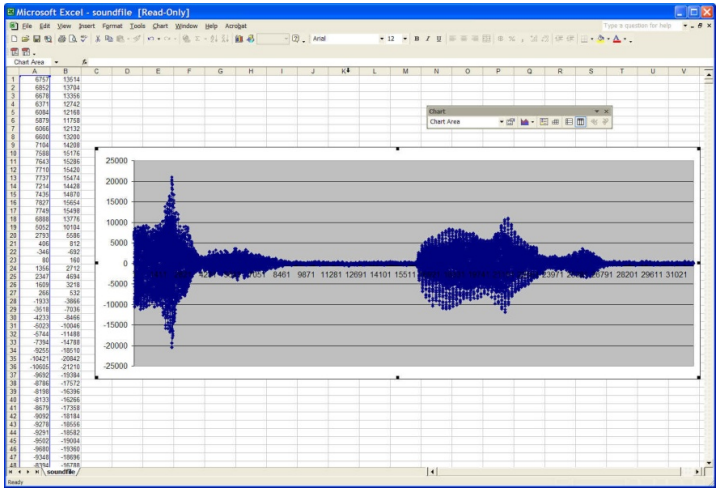
- What this leaves us with is a long file, containing just numbers.
- What knows how to deal with long lists of numbers?
  - **Excel**!
- We can simply open our text (.txt) file in Excel.



	A
1	6757
2	6852
3	6678
4	6371
5	6084
6	5879
7	6066
8	6600
9	7104
10	7588
11	7643
12	7710
13	7737
14	7214
15	7435
16	7827
17	7749
18	6888
19	5052
20	2793
21	406
22	-346
23	80
24	1356
25	2347
26	1609
27	266
28	-1933
29	-3518
30	-4233
31	-5023

We Can Process the Sound in Excel

- We can graph the sound (below)
  - A signal view is simply the graph of the sample values!
- We can add a column and do some modification to the original sound. (Fill down to get them all.)
  - Can increase the volume that way.



Reading the Text Back as a Sound

- **while** – keeps executing the block until the logical expression is **false**.
- **(soundIndex < getLength(sound))** – while the index is not yet at the end of the sound, so there's still room for more numbers.
- **and** – both parts have to be true for the whole thing to be true.
- **(fileIndex < len(contents))** – while there are any numbers left in the file, i.e., the **fileIndex** is before the length of the contents of the file.

```
def textToSound(filename):
  #Set up the sound
  sound = makeSound(getMediaPath("sec3silence.wav"))
  soundIndex = 1
  #Set up the file
  file = open(filename,"rt")
  contents=file.readlines()
  file.close()
  fileIndex = 0
  # Keep going until run out sound space or run out of file contents
  while (soundIndex < getLength(sound)) and (fileIndex < len(contents)):
    sample=float(contents[fileIndex])  #Get the file line
    setSampleValueAt(sound,soundIndex,sample)
    fileIndex = fileIndex + 1
    soundIndex = soundIndex + 1
  return sound
```

We Could do Pictures, but More Complicated

- Pictures aren't just a single number for each pixel
- To recreate a picture in text we need to record, for each pixel:
  - The X and Y positions
  - The R ,G, and B component values
- That requires more structured text than simply a long line of numbers.
- Let's do that in just a few minutes.

Mapping from Text to Anything

- Once we've converted to text (or numbers), we can do anything we want.
- Like, mapping from sound to…pictures!

We Simply Decide on a Representation: How Do We Map Sample Values to Colors?

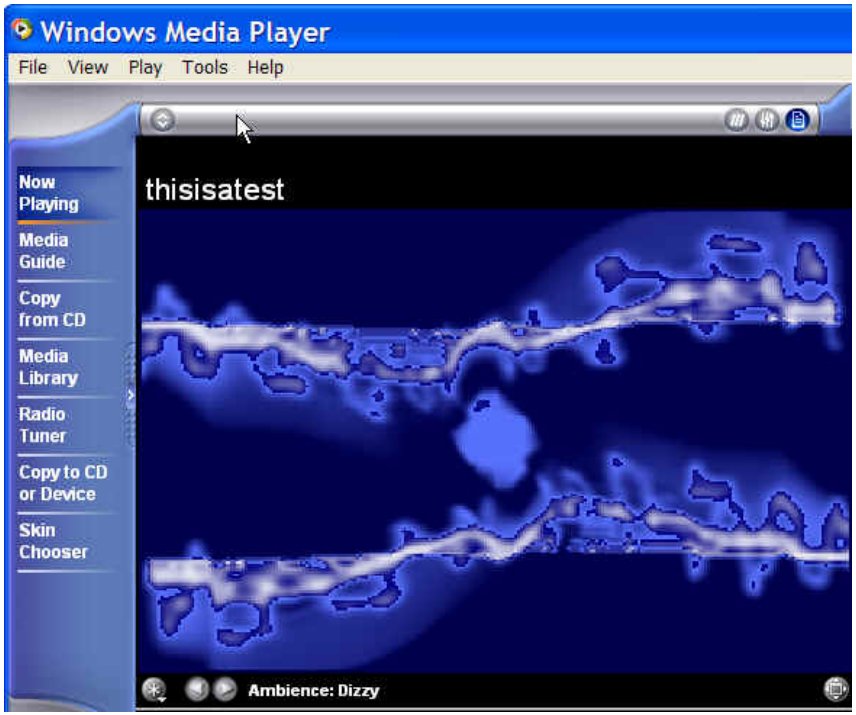
Here's one:

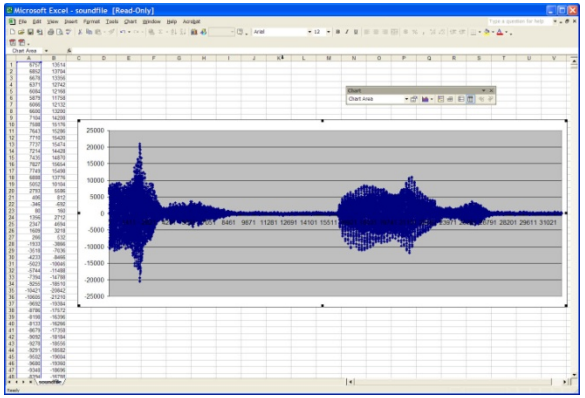
- Greater than 1000 is red
- Less than 1000 is blue
- Everything else is green

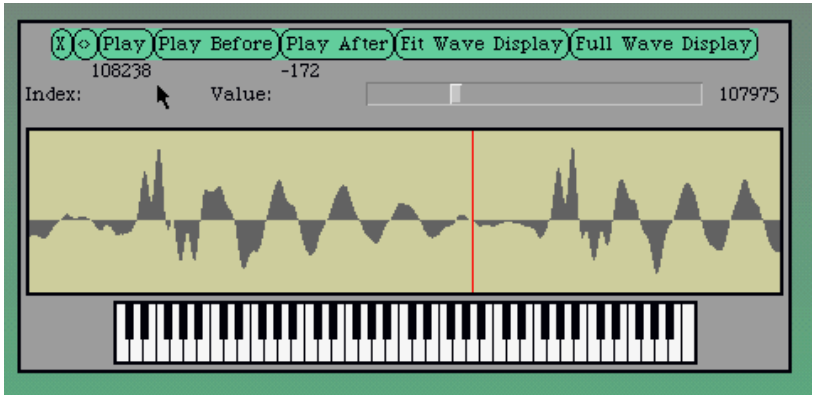
```
def soundToPicture(sound):
  picture = makePicture(getMediaPath("640x480.jpg"))
  soundIndex = 0
  for p in getPixels(picture):
    if soundIndex == getLength(sound):
      break
    sample = getSampleValueAt(sound,soundIndex)
    if sample > 1000:
      setColor(p,red)
    if sample < -1000:
      setColor(p,blue)
    if sample <= 1000 and sample >= -1000:
      setColor(p,green)
    soundIndex = soundIndex + 1
  return picture
```

- **break** is yet another new statement.
- It literally means "Exit the current loop."
- It's most often used in the block of an **if**
  - "If something extraordinary happens, leave the loop immediately."
- In this case, "If we run out of samples before we run out of pixels, STOP!"

Any Visualization of Sound is Merely an Encoding







Any Visualization of Any Kind is Merely an Encoding

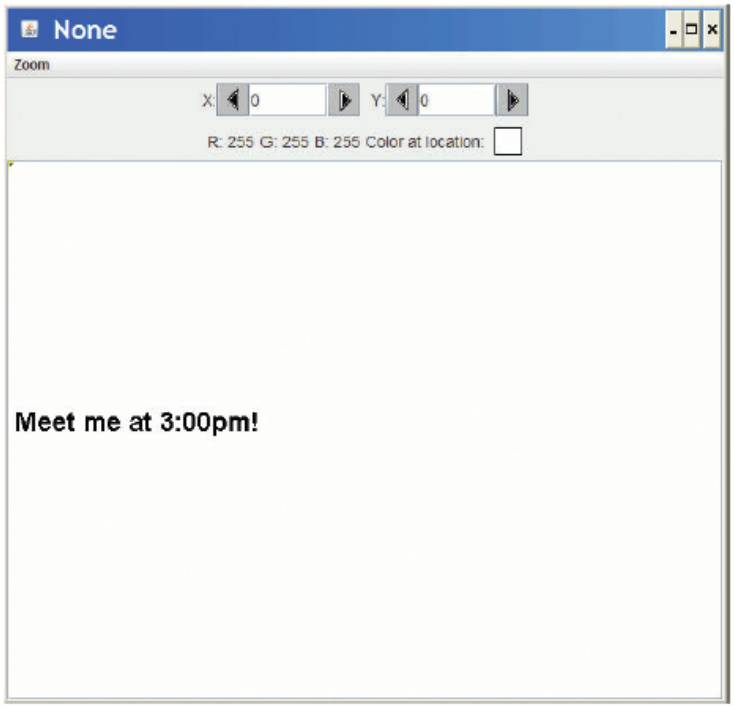
- A line chart? A pie chart? A scatterplot?
  - These are just lines and pixels set to correspond to some mapping of the data
- Sometimes data is lost
  - "Lossy" compression being used?, or...
  - Going from RGB color → grayscale
- Sometimes data is not lost, even if it looks like a dramatic change.
  - A negative of an image, then taking the negative of a negative to get back to the original.

Hiding Text in a Picture

- **Steganography** is hiding information in ways that can't be easily detected.
- One form of steganography is hiding text information of a picture.

Our Algorithm for Hiding Text

- We'll draw our message in black pixels on a message picture.
- We'll hide our message in a picture of the same size.
- First: Make sure that all red values are **even**.
- Second: For every pixel where the message picture is black, add one to the red value at the corresponding x,y.

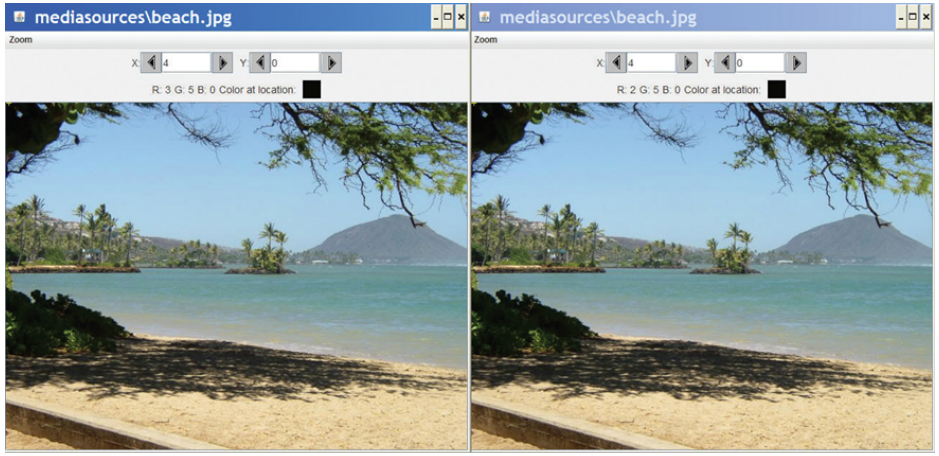


Function to Encode the Message

```
def encode(msgPic ,original ):
    # Assume msgPic and original have same dimensions
    # First , make all red pixels even
    for pxl in getPixels(original ):
            # Using modulo operator to test oddness
            if (getRed(pxl) % 2) == 1:
                    setRed(pxl , getRed(pxl) - 1)
    # Second , wherever there ' s black in msgPic
    # make odd the red in the corresponding original pixel
    for x in range(0, getWidth(original )):
            for y in range(0, getHeight(original )):
                    msgPxl = getPixel(msgPic ,x,y)
                    origPxl = getPixel(original ,x,y)
                    if (distance(getColor(msgPxl),black) < 100.0):
                            # It' s a message pixel! Make the red value odd.
                            setRed(origPxl , getRed(origPxl )+1)
```

Doing the Encoding

```
>>> beach = makePicture(getMediaPath("beach.jpg"))
>>> explore(beach)
>>> msg = makePicture(getMediaPath("msg.jpg"))
>>> encode(msg,beach)
>>> explore(beach)
>>> writePictureTo(beach,getMediaPath("beachHidden.png"))
```



Original Encoded

It's really important to save the message as .PNG or .BMP, **not** JPEG. JPEG is **lossy** so pixel color values might change. PNG and BMP are lossless formats.

Decoding: Getting the Message Back

- Create a new “message” picture of same size as the encoded image.
- For each pixel, if the red value is **odd**, make the pixel in the message at the same x,y **black.**

```
def decode(encodedImg):
  # Takes in an encoded image. Return the original message
  message = makeEmptyPicture(getWidth(encodedImg),getHeight(encodedImg))
  for x in range(0,getWidth(encodedImg)):
    for y in range(0,getHeight(encodedImg)):
      encPxl = getPixel(encodedImg,x,y)
      msgPxl = getPixel(message,x,y)
      if (getRed(encPxl) % 2) == 1:
        setColor(msgPxl,black)
  return message
```

Encoding Sound in a Picture

```
def encodeSound(sound,picture):
  soundIndex = 0
  for p in getPixels(picture):
    # Clear out the red LSB
    r = getRed(p)
    if ((r % 2) == 1):
      setRed(p,r-1)
  for p in getPixels(picture):
    # Did we run out of sound?
    if soundIndex == getLength(sound):
      break
    # Get the sample value
    value = getSampleValueAt(sound,soundIndex)
    if value > 0:
      setRed(p,getRed(p)+1)
    soundIndex = soundIndex + 1
```

Decoding Sound from a Picture

```
def decodeSound(picture):
  sound = makeEmptySoundBySeconds(5)
  sndIndex = 0
  for p in getPixels(picture):
    # Did we run out of sound?
    if sndIndex == getLength(sound):
      break
    # Is it mostly red, mostly blue, or mostly green?
    if ((getRed(p) % 2) == 1):
      setSampleValueAt(sound,sndIndex,32000)
    else:
      setSampleValueAt(sound,sndIndex,-32000)
    sndIndex = sndIndex + 1
  return(sound)
```