Frequently-asked Questions for Assignment #3

1. Q: How do I get the words that the user typed as command-line arguments?

   A: You could join the words one at a time, with a space between them:

   ```
   text = ''
   for word in sys.argv[1:]:
       text += word + ' '
   ```

   Or you can let python do it for you:

   ```
   text = ' '.join(sys.argv[1:])
   ```

2. Q: How do I create a two-dimensional array?

   A: It's just an array of arrays:

   ```
   picture = []
   for row_index in range(height):
       picture.append([" "] * width)
   ```

3. Q: I'm confused. Why should I make an array of arrays? Can't I just make an array of strings?

   A: You could, but it's a bit awkward to replace one character in a string, because strings are immutable. You would have to make a fresh string using slices:

   ```
   picture[y] = picture[y][:x] + symbol + picture[y][x+1:]
   ```

   whereas it's easier with an array of arrays:

   ```
   picture[y][x] = symbol
   ```

4. Q: That's weird. `picture[y][x]` ? Why not `picture[x][y]`?

A: `picture` is an array of arrays. `picture[y]` is one row in that array. `picture[y][x]` is one entry in that row. The `[]` parentheses work left-to-right.

5. Q: How do I get the blocks in the first place?

   A: They are in the file `matrix_5x7.txt` . Open that file, and read its lines. Some lines are 1-character symbols, and other lines are 5-character rows in a picture.

   Each block uses up 8 lines of the file: a symbol, plus the 7 lines that make the block. You can assemble them thus:

```
all_lines = handle.readlines()
for index in range(0, len(all_lines), 8):
    symbol = all_lines[index].rstrip("\r\n")
    line1  = all_lines[index + 1].rstrip("\r\n")
    line2  = all_lines[index + 2].rstrip("\r\n")
    ...

handle.close()
```

6. Q: OK, I got the symbol, and 7 strings. How do I store these?

   A: This is a perfect case for using a dictionary:

```
        block: List[str] = [line1, line2, line3, line4,
                            line5, line6, line7]
        matrices[symbol] = block
```

7. Q: Will this work?

```
    for i in range(len(text)):
        letter = text[i]
        ... get the block for this letter ...
        ... print the block ...
```

   or similarly:

```
    letters = list(text)
    for letter in letters:
        ... get the block for this letter ...
        ... print the block ...
```

   A: No. If the text is `"ABC"`, you will get

```
  X
 X X
X   X
X   X
XXXXX
X   X
X   X

XXXX
X   X
X   X
XXXX
X   X
X   X
XXXX

 XXX
X   X
X
X
X
X   X
 XXX
```

You can't print a line, until you have gathered all the blocks for that line. You need to create a big rectangular picture, an array of "pixels", some of which are spaces, some are `X`. **Then** you should print the whole picture.

8. Q: If I have a 5 x 7 block, how do I figure out where its pixels are, in the picture?

   A: If you know the `text_row`, `text_col` of your block, you can use math. Each block is 7 wide, and 8 high (5+2, 7+1).

   You can figure out the block's top-left pixel:

```
top_left_x = text_col * 7
top_left_y = text_row * 8
```

   Now, for each entry `x y` in the 5 x 7 block, you should add to `top_left_x` and `top_left_y` to get the entry's `x` and `y` coordinates:

```
pixel_x = top_left_x + x
pixel_y = top_left_y + y
```

9. Q: I'm confused by all those widths and heights.

   A: Remember that you are dealing with letters, *and* also with pixels.

`text_height` and `text_width` are how many rows and columns your text takes up.

Example: if the message is `"Hello You"`, then `text_height` is 1, and `text_width` is 9.

Example: if the message is `"Hello World byebye"`, then `text_height` is 2, and `text_width` is 10 (there are two rows of text, at most 10 letters wide).

The `picture` is made up of pixels. Each letter takes up 5 pixels wide, plus 2, so 10 letters take up 7*10 + 6 = 76 pixels wide. So your picture should be 76 pixels wide.

Each row of text takes up 7 rows of pixels, plus 1, so 3 rows of text would take up 2*7 + 6 = 20 rows of pixels. So, in that case, your picture would be a 2-dimensional array. It has 20 entries, and each entry is an array of 76 `" "` spaces.