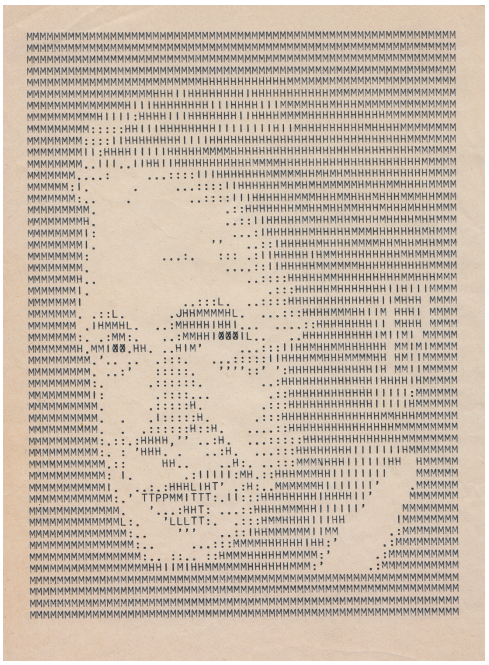
INDU: ASCII Art Studio

Lars Arvestad

Innehåll

Introduction

The art form *ASCII art* originates from the early days of computers and involves expressing oneself visually through text files. These could be displayed on text terminals and sent to simple printers without graphic capabilities. See Figure 1 below for an example.



Figur 1: Dag Hammarskjöld as ASCII art. Photo: Jonn Leffmann, CC BY 3.0, via Wikimedia Commons.

ASCII is a character encoding that was launched in the 1960s and can still be found today. The principle of ASCII art is that completely white pixels are represented by a space, black pixels by a character with "a lot of ink" such as #, and shades of gray are represented by other characters. Figure 2 shows how grayscale can be represented with ASCII art.

```
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
.....::::ijjjjxxxxoooOhhhhXXXX%@@@@#########
```

Figur 2: A grayscale in a JPG picture converted to ASCII art.

In this project, you will write a program, *ASCII Art Studio*, where users can interactively experiment with the art form and automatically render image files into ASCII. Your program should be able to read, for example, JPG and PNG images, and with a simple command language, users should be able to handle multiple images simultaneously, adjust contrast and exposure of the images, and experiment with different output sizes.

The project has two tasks at different levels. The first task is simpler and can at most result in a grade of C. The second level can result in a grade of A, but requires more implemented functionality and also the use of object orientation for implementation. If you choose the A level, you do not need to submit a separate solution for task 1.

Aids

To handle image files, we recommend the `pillow` module (Python Image Library). There is ample online documentation for `pillow`, including technical reference documentation and general tutorials with good examples. As a supplement, we list some operations that are useful to use.

For task 2, it is good to use a module for serialization (i.e., transforming a data structure or class into a textual representation in a file). We recommend `json`, but there are others that can also be used, such as `pickle`.

For the command language, you may use the `re` module if you wish. Note that it may be challenging to use if you are not familiar with regular expressions.

You may also use `math` if it makes things easier.

Open an image file

To open an image file and read the image to a Python object, you can write like this:

```
with Image.open(filename) as img:
    img.load()
```

Afterwards, various operations can be performed on the image, including displaying it in a separate window. Pillow supports several graphic formats, including JPG and PNG.

Convert to grayscale

It becomes easier to transform pixel images into ASCII art if you convert color images to grayscale. This can be done easily with the following conversion instruction.

```
new_img = img.convert(mode='L')
```

Pixels

By reading the grayscale level of a pixel, you obtain a numerical value that can be converted into a letter.

```
pos = (x, y)
grey = img.getpixel(pos)
```

Note that the method `getpixel` takes one argument, which should be a pair. In the example above, x and y are the coordinates for some point in the image.

Task 1 (E-C)

Write a program where you can enter commands to work with ASCII art. The commands to be implemented are:

load *filename* Load the graphic file *filename* and make it the "current image". If an error occurs during loading (file does not exist, incorrect filename, unreadable file, wrong format, etc.), an error message should be displayed.

render Take the current image and print ASCII art representing the image. The width of the printed image should be 50 characters. The height should be chosen so that the proportions of the image are preserved and adjusted for the fact that the rectangles used by fonts tend to be narrow (and font-dependent). Note that your console (in Spyder or equivalent) should use a font with fixed-width characters (such as Courier).

info Print information about the current image: filename and size. If no image has been loaded, the text "No image loaded" should be printed.

quit End the session.

Example run

In the example below, `AAS` is the program's prompt. The instructions are:

- `load slalom.jpg`
- `render`
- `quit`

Can you see the skier?

```
Welcome to ASCII Art Studio!
AAS: load slalom.jpg
AAS: render

      ,ii.
     :&%@&,..
     :&%%@%X%&i:::.
     .o%@@@@@@@@@%O.
    .&@@@%@@@@@&&%%:
 .  :%@@@@@@@@@%mij.,,,,,,,,....................
... ,X@@@@@@@@@@%m.........,,,,,,,,......    ....
.....X#&jO@@@@@@%Om:....................., ........
.....O#o .o&OXX&oooxi...             ...........
.  . i@m  .:ixxmiixxxmmnnj:::,   ..,,....
     j%&,    :inmxmmx&Ommxminxon,,::,,,,,.    ....
      ,,...   .,jnmxxOm::iiijjj:j:::::,. .
           ....:jinniOXOxnj:,,...
             .:omij,,:j,.


AAS: load grayscale.jpg
```

```
AAS: render
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
       ....::::ijjjjxxxxoooOhhhhXXXX%@@@@########
AAS: quit
Bye!
```

Requirements for Task 1

- Your ASCII art should be a reasonable representation of the original image. You can choose how grayscale is converted into letters, but it should be recognizable as ASCII art.
- The printed ASCII art should have a width of 50 characters.
- It should be possible to input commands and display images.

Task 2 (E-A)

In this task, we increase the number of commands and the functionality that needs to be supported. Additionally, you *must* use object-oriented programming to structure your code.

Extended functionality

- The program should support working with multiple images. In a *session*, the loaded images should be accessible, either under the filename or an alias, and you should be able to display the one you wish. A session should then be able to be saved to be loaded again later. What is saved is the filenames of the images you have loaded and their associated information, but not the image data (pixels). If you load a saved session, you should be able to continue working with the old images.
- Each session has a current image, *current*, which is the image you last used. Every time you refer to an image (with load, render, set, and so on, see below), current is set to that image. This makes commands easier to write in this way.
- You should be able to determine some properties of the images:
  - *target size*, which is the size of the ASCII art to be printed.
  - *brightness*, which makes the image lighter or darker.
  - *contrast*, which enhances the difference between light and dark.

Tip: For *brightness* and *contrast*, there are methods in Pillow that can be used.

- The command language is expanded to support the new features, see below.

Command language

The commands that should be supported are:

load image *filename* Load the file in *filename* and save it in the session under its filename. The most recently loaded image is also `current`.

load image *filename* as *alias* Same as above, but save the image under the name *alias*.

info List the loaded images and their properties, as well as which image `current` points to.

render Create ASCII art for the current image. When the command is given, the image should be given the correct size, contrast, and brightness. The width of the rendered image should be 50, if not specified otherwise (see below). The height should be chosen so that the proportions of the image are preserved and adjusted for the fact that the rectangles used by fonts tend to be narrow (and font-dependent).

render *img* Same as render, but for the image saved under the name *img*. It can be the filename, its alias, or `current`.

render *img* to *filename* Same as above, but the output is saved to a file.

set *img* width *num* Specify that the width of the image *img* (alias or filename) should be *num*. The height of the image should be adjusted so that the aspect ratio height/width is maintained. This size should be used when rendering the image.

set *img* height *num* Same as above, but for the height of the image.

set *img* brightness *num* Specify how the brightness of the image *img* should change compared to the original before rendering. If *num* is 1.1, the image should become 10 % brighter, and if *num* is 0.8, the image should become 20 % darker.

set *img* contrast *num* Same as above, but for the contrast.

save session as *filename* Save the loaded images as filenames, along with their size, brightness, and contrast. The image pixels are not saved. The current image, `current`, is also saved.

load session *filename* Load the session saved in *filename*. The images whose filenames are saved should be loaded again, and the saved parameters should be set. The current image should be the one specified in the saved session.

quit End the session.

Example run

```
Welcome to ASCII Art Studio!
AAS: load image slalom.jpg as slalom
AAS: load image stadshuset.jpg as hus
AAS: info
=== Current session ===
Images:
slalom
   filename: slalom.jpg
   size: (728, 485)
   target size: (50, 20)
   brightness: 1.0
   contrast: 1.0
```

```
hus
   filename: stadshuset.jpg
   size: (640, 426)
   target size: (50, 20)
   brightness: 1.0
   contrast: 1.0

Current image: hus
AAS: save session as s1
AAS: quit
Bye!
```

If we restart ASCII Art Studio, we should be able to continue with these images:

```
Welcome to ASCII Art Studio!
AAS: load session s1
AAS: render
xxxxxxxxxxxxxxxxxxxxxxxnxnnnnnnnnnnnnnnnnnnnnnnnnn
xxxxxxxnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnnnnnnnnnxxnnnnnnnnnnnjjjjjjjjjjjnn
nnnnnnnnnnnnnnnnnnnnnjx&Xxjjjjjjjjjjjjjjjjjjjjjjj
nnnnnnnnnnjjjjjjjjjjjjjnoonjjjjjjjjjjjjjjjjjjjjjj
nnnnnnnnnjjjjjjjjjjjjjjo&XOjjjjjjjjjjjjjiiiiiiiii
nnjjjjjjjjjjjjjjjjjjjjhXX@@&jjjjjjjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjjjh&&@#%jjjjjjjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjjj&&&@#%jjjjjjjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjij&&&@#%jiiiiiiiijjjjjjjjjjjj
jjjjjjjjjjjiiiiiiiiij&&&@#@jiiiiiiiiiiiiiiiiijj
jjjjjjijjiiiiiiijiiij&&&@#@jiiiiijiiiiiiiiiiii
jnnnjnjnonnnnxnxoxxxoo&&&@@@&hhOonj&jiiiiiiiiiji
oOhhoOOoh&hhhhhhhhh&&&&h&@@#####%X&Xhxiiiijjjjjji
%%%%XX&hXhhhhhhhhhhhhhh&&%@########%%%X&ojjji::iin
%%%@%X&h&hhO000000000hhh&h%@@#######X%@@%%hOh::ixxx
%@@@X%X&&&&h&&&&&&&&h&&h&%%%@###@@#@X%%X&&&&hOOxh&&
XXX&&Xhhh&&&&&&&&&&&&X%X%@@%%%%XX%XX&&&h0Oo0OOoh&&
hhhh0000h00000000000hh&XX%%X&hhh00000ooooxxxxxxxxx
00000oo000000000000000h&XX%X&h000000000000000o0oooo
AAS: set hus width 80
AAS: render
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxnxnnxxnxnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxnnnnnxnnnnnnnnxnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
xxxxxxxxxxxnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
xxxnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnxxnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnxonxonnnnnnnnnnnjjjjjjjjjjjjjjjjjjjjjjjjjjjj
nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnxX@@Xnjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
nnnnnnnnnnnnnnnjjjjjjjjjjjjjjjjjjjjjjjohOojjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
nnnnnnnnnnnnnnnnnnnnnnnnjjjjjjjjjjjjjjooxojjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
nnnnnnnnnnnnnnnnjjjjjjjjjjjjjjjjjjjjjoh&&Xxjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjiiiiiiiii
nnnnnnnnnnjjjjjjjjjjjjjjjjjjjjjjijnxhXX%@hxnjjjjjjjjjjjjjjjjjjiiiiiiiiiiiiiiiii
njjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjhXX&&@###hjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjO&&&&@###hjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjh&&&&@@@#&jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjh&&&&@@@#&ijjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjh&&&&@#@#&ijjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjiiiijhX&&&@#@#Xiiiiiiiiiiiiiiiiiijjjjjjjjjjjjjjjj
jjjjjjjjjjjjjjjjjjjiiiiiiiiiiiiiiihX&&X@#@#Xiiiiiiiiiiiiiiiiiiiiiiiiiiiijjjj
jjjjjjjjjjjjjjjiiiiiiiijjiiiiiiihX&&&@#@#Xiiiiiiiiiiiiiiiiiiiiiiiiiiiiij
```

```
jjjjjjjjjjjjjjiiiiiiiiiiiijjiiiiiiihX&&&@#@#%iiiiiiiiiiijjiiiiiiiiiiiiiiiiiiii
jjjjjjjjjjiinOiiiiiiijiijnxjjnjnnnx&&&&&@#@#%xnnnjjjjiijhoiiiiiiiiiiiiiiiiiijjii
jnxooxoinojxxhoxoooo0Ooo000o0000000&&&&&@#@@###@@@%XhOnnXhiii:iiiiiiijjjiiiijjji
ooOOOOhjo&oOxh&h0000h0h0hhhhhh&&&&&&hh&%@@@######@@X&hhXXOo::iiiiijjjjjjjjjjii
&&%XXXX&&&&Oh%X&&&&&&&&&&&&&&&&&&h&&h&&&%@@##############@X%%&Oonjiijjjjnnjijjjjj
%%%%%%%XXX&h&XOh0000h000000000hhhhhhhXXh%@@#@############%X%%%%%hxnjjjj:,.::n:xx
%%%@%%@@&%&hhXhh&h00000000000h000h0hh&&h%@@#@############XX@@@@%@%Xooohn ,,njOiox
%@%@@@@XX@&hhXOhh000000000000000hhhhh&&h%@@@%############&X%%%%%XX&&&&Ojxiiohо0x
X@@#%#%&X@XhXX&&&&&&&XXXXXXXXX&&hXhhhh&X@X&%@###@#@@@@@XX%%&XX&&&&&hhhhhOoO&h&&
%%X%XX&XX%hoh0h&&&&hh&&&&X&&&&&h&hX%%X%@%@@@%%%@@@%%@@%X%%%XXXhh000oo0h000o0#%%@
&X&&X&&&&&hhhhh&&h&&&&&&&&&hh&&&&&XX%%%%@@@%%XXXXXX&&&&&&&hhh000000000000oo00000
hhhhhhh0000000000000000000000000hh&XXX%%@%%X&hhh000000000ooooooxxxxxxxnnnnxxxxx
00000000000000000000000000000000000hh&&&XX%%XX&hhh000000000000ooooooooooxxxxxxxx
0000000ooooooOoooo0oo000000000000000hh&XXX%%%X&h0000000000000000000000000000000O
AAS: quit
Bye!
```

Requirements for Task 2

- Your program should implement the command language as specified above and in the example run.
- You should use object-oriented programming in a meaningful way to score points for program structure.

Grading

Details can be found with the general instructions in *INDU—Individual Assignment in Programming*.