# INDU: ASCII Art Studio

### Lars Arvestad

## Innehåll

Hjälpmedel		 	 	•	 	 •	•	 		 					•	
Uppgift 1 (E-C)																
Exempelkörning																
Krav på uppgift 1		 	 		 	 •		 		 						
Uppgift 2 (E-A)																
Utökad funktionalitet		 . <b>.</b> .	 		 			 		 						
Kommandospråket		 	 		 			 		 						•
Kommandospråket Exempelkörning Krav på uppgift 2	• •	 	 		 			  		 		:		 		

## Introduktion

Konstformen ASCII art är från datorernas barndom och den går ut på att uttrycka sig visuellt med textfiler. Dessa kunde visas på textterminaler och skickas till enkla skrivare utan grafikmöjligheter. Se figur 2 för ett exempel. ASCII är den teckenkodning som lanserades på 60-talet och kan återfinnas än idag. Principen med ASCII art är att helt vita bildpunkter representeras av ett mellanslag, svart av något tecken med "mycket trycksvärta", till exempel M, och låter grånivåer ges av andra tecken. Figur 1 visar hur en gråskala kan representeras med ASCII art.

I det här projektet ska du skriva ett program, *ASCII Art Studio*, där man kan experimentera med interaktivt med konstformen, men med automatisk rendering av bildfiler till ASCII. Ditt program ska kunna läsa till exempel JPG- och PNG-bilder och med ett enkelt kommandospråk ska användaren kunna hantera flera bilder samtidigt, justera kontrast och exponering av bilderna, och experimentera med olika storlekar på utdata.

Projektet har två uppgifter på olika nivå. Den första är enklare och kan som högst ge dig betyg C. Den andra nivån kan ge betyg A, men kräver mer implementerad funktionalitet och att du dessutom använder objektorientering för implementationen. Om du bestämt dig för A-nivån behöver du inte lämna in en lösning för uppgift 1.

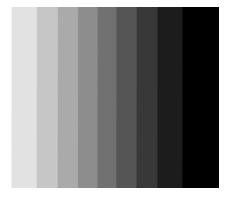
## Hjälpmedel

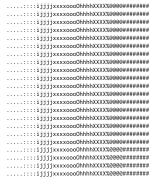
För att hantera bildfiler rekommenderar vi modulen pillow (*Python Image Library*). Det finns bra med dokumentation för pillow, både teknisk referensdokumentation och allmän handledning med bra exempel. Som komplement listar vi några operationer som är bra att använda.

För uppgift 2 är det bra att använda en modul för serialisering (dvs, att transformera en datastruktur eller klass till ett textrepresentation på fil). Vi rekommenderar json, men det finns andra som också får användas, som till exempel pickle.

För kommandospråket får du använda modulen re om du vill. Observera att den antagligen är jobbig att använda om man inte lärt sig om regexp sen tidigare.

Du får också använda math.





Figur 1: Gråskala, från JPG och konverterat till ASCII art.

### Öppna en bildfil

För att öppna en bildfil och läsa in bilden till ett objekt i Python kan man skriva så här:

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

Därefter kan man göra diverse operationer på bilden, inklusive att visa upp den i ett eget fönster. Pillow stödjer ett flertal grafikformat, inklusive JPG och PNG.

### Konvertera till gråskala

Det blir lättare att transformera pixelbilder till ASCII art om man konverterar färgbilder till gråskala. Det görs enkelt med följande konverteringsinstruktion.

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

#### **Pixlar**

Genom att läsa av grånivån på en bildpunkt, en pixel, får man ett siffervärde som kan konverteras till en bokstav.

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

Observera att metoden getpixel tar ett argument och det ska vara ett par. I exemplet ovan är x och y koordinaterna för någon punkt i bilden.

# Uppgift 1 (E-C)

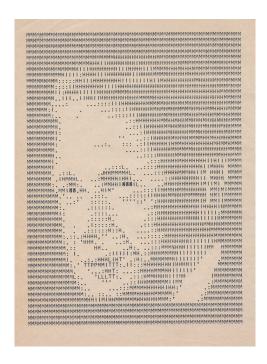
Skriv ett program där du kan skriva kommandon för att arbeta med ASCII art. De kommandon som ska implementeras är:

**load** *filename* Läs in grafikfilen *filename* och gör den till "aktuell bild". Om det uppstår ett fel vid inläsningen (filen finns inte, fel filnamn, oläsbar fil, fel format, osv) så ska ett felmeddelande skrivas ut.

render Tag aktuell bild och skriv ut ASCII art som representerar bilden. Bredden på den utskrivna bilden ska vara 50 tecken. Höjden bör väljas så att bildens proportioner bevaras och med en justering för att de rektanglar som fonter använder tenderar att vara smala (och fontberoende). Observera att din konsol (i Spyder eller motsvarande) bör använda en font med fix bredd på tecken (som till exempel Courier).

**info** Skriv ut information om aktuell bild: filnamn och storlek. Om ingen bild har lästs in så ska texten "No image loaded" skrivas ut.

quit Avslutar sessionen.



Figur 2: Dag Hammarskjöld som ASCII art. Foto: Jonn Leffmann, CC BY 3.0, via Wikimedia Commons.

# Exempelkörning

• load slalom.jpg

I exemplet nedan är AAS programmets prompt. Instruktionerna är:

```
• render
   • quit
Ser du slalomåkaren?
Welcome to ASCII Art Studio!
AAS: load image slalom.jpg
AAS: render
     ,ii.
    : &%@&, . .
    :&%%@%X%&i:::.
     .0%@@@@@@@@@%O.
    . &@@@%@@@@@&&%%:
   :%@@@@@@@@mij.,,,,,,,....
...,X@@@@@@@@@m....,,,,,,,,,
.....X#&j0@@@@@%Om:...., ...., ....
.....0#o .o&OXX&oooxi...
  . i@m .:ixxmiixxxmmnnj:::, ..,,....
           :inmxmmx&Ommxminxon,,::,,,,.
    j%&,
             .,jnmxxOm::iiijjj:j::::,. .
          ....:jinniOXOxnj:,,...
            .:omij,,:j,.
AAS: load grayscale.jpg
AAS: render
```

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

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

AAS: quit Bye!

## Krav på uppgift 1

- Din ASCII art ska vara en rimlig representation av originalbilden. Du kan själv välja hur gråskala ska konverteras till bokstäver, men det ska gå att se att det är ASCII art.
- Utskriven ASCII art ska ha bredd 50 tecken.
- Det ska gå att ge kommandon och visa bilder

# Uppgift 2 (E-A)

I denna uppgift ökar vi antalet kommandon och den funktionalitet som ska stödjas. Dessutom ska du använda objektorientering för att strukturera din kod.

### Utökad funktionalitet

- Programmet ska stödja att man arbetar med flera bilder. I en session ska de inlästa bilderna finnas åtkomliga, under filnamnet eller ett alias, och man ska kunna visa den man önskar. En session ska sedan kunna sparas för att läsas in på nytt senare. Det som sparas är filnamnet på de bilder man har läst in och deras associerade information, men inte bilddata (pixlarna). Om man läser in en sparad session så ska man kunna fortsätta jobba med de gamla bilderna.
- · Varje session har en aktuell bild, current, som är den bild som man senast använde. Varje gång man refererar till en bild (med load, render, set, och så vidare, se nedan) så sätts current till den bilden. Kommandon ska bli enklare att skriva på det viset.
- Du ska kunna bestämma några egenskaper på bilderna:
  - target size, som är storleken på den ASCII art som skrivs ut.
  - brightness, som är gör bilden ljusar eller mörkare.
  - contrast, som förstärker skillnaden på ljust och mörkt.

För brightness och contrast finns det metoder i Pillow som kan användas.

• Kommandospråket utökas till att stödja de nya finesserna, se nedan.

## Kommandospråket

De kommandon som ska stödjas är:

**load image** *filename* Läs in filen i *filename* och spara den i sessionen under dess filnamn. Senast inlästa bild är också current.

load image filename as alias Som ovan, men spara bilden under namnet alias.

info Lista inlästa bilder och deras egenskaper, samt vilken bild som current pekar på.

render Skapa ASCII art för aktuell bild. När kommandot ges ska bilden ges rätt storlek, kontrast, och ljusstyrka. Bredden på renderad bild ska var 50, om annat ej angetts (se nedan). Höjden bör väljas så att bildens proportioner bevaras och med en justering för att de rektanglar som fonter använder tenderar att vara smala (och fontberoende).

**render** *img* Som render, men för den bild som är sparad som med namnet *img*. Det kan vara filnamnet, dess alias, eller current.

render img to filename Som ovan, men utdata sparas på fil.

**set** *img* width *num* Bestäm att bredden på bilden *img* (alias eller filnamn) ska vara *num*. Bildens höjd ska justeras så att proportionen höjd/bredd bibehålls. Det är denna storlek som ska användas när man renderar bilden.

set img height num Som ovan, men för bildens höjd.

**set** *img* **brightness** *num* Ange hur ljusstyrkan för bilden *img* ska ändras jämfört med originalet innan renderas. Om *num* är 1.1 så ska bilden bli 10% ljusare, och om *num* är 0.8 så ska bilden bli 20% mörkare.

set img contrast num Som ovan, men för kontrast.

**save session as** *filename* De inlästa bilderna sparas som filnamn, storlek, ljuststyrka, och kontrast. Bildens pixlar sparas inte. Aktuell bild, current, sparas också.

**load session** *filename* Läs in den session som är sparad i *filename*. De bilder vars filnamn är sparade ska läsas in på nytt och sparade parametrar ska sättas. Aktuell bild ska vara den som anges i den sparade sessionen.

quit Avslutar sessionen.

### Exempelkörning

I följande exempel läser vi in bilderna som är givna i lasse.jpg och stadshuset.jpg

```
Welcome to ASCII Art Studio!
AAS: load image lasse.jpg as lasse
AAS: load image stadshuset.jpg as hus
AAS: info
=== Current session ===
Images:
lasse
   filename: lasse.jpg
   size: (2458, 3565)
   target size: (50, 44)
  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!

Om vi återstartar ASCII Art Studio, så ska vi kunna fortsätta med dessa bilder:

Welcome to ASCII Art Studio!

AAS: load session s1

AAS: render

nnnnnnnnnnnnnnnnnnnnix&Xxjjjjjjjjjjjjjjjjjjjjjjjjj nnnnnnnnjjjjjjjjjjjjjo&XOjjjjjjjjjjjjjjiiiiiiiiii jjjjjjjjjjjjjjjjh&&@#%jjjjjjjjjjjjjjjjjjjjjjjjjj jjjjjjjjjjjjjjjjk&&@#%jjjjjjjjjjjjjjjjjjjjjjjjjj jjjjjjjjjjjjjjjij&&&@#%jiiiiiiiijjjjjjjjjjjjjjj jnnnjnjnonnnnxnxoxxxoo&&&@@@&hhOonj&jiiiiiiiiiiiji oOhhoOOoh&hhhhhhhhhhk&&&h&@@#####%X&Xhxiiiijjjjjjii %%%@%X&h&hh00000000hhh&h%@@######X%@@%%hOh::ixxx %@@@X%X&&&&h&&&&&&Ah&&h&%%%@##@@#@X%%X&&&&hOOxh&& XXX&&Xhhh&&&&&&&&&&X%X%@@%%%XXXXX&&&hOOoOOoh&& hhhh0000h000000000hh&XX%X&hhh0000oooxxxxxxxxx 00000oo00000000000000h&XX%X&h00000000000000oooo

AAS: set hus width 80

AAS: render

xxxxxxxxxxxxxxxxxxxxxxxxjjjjjjjjiinOiiiiiiijiijnxjjnjnnnx&&&&@#@#%xnnnjjjjiijhoiiiiiiiiiiiiiiiiiijjii jnxooxoinojxxhoxoooo00oo000o000000000&&&&&@#@@###@@@%XhOnnXhiii:iiiiiijjjiiijjji ooOOOOhjo&oOxh&hOOOOhOhOhhhhhhhk&&&&&&&hh&%@@@######@@X&hhXXOo::iiiiijjjjjjjjjjiii

# Krav på uppgift 2

- Ditt program ska implementera kommandospråket som specificerat i sektion och exempelkörningen.
- Du ska använda objektorientering på ett meningsfullt sätt för att få poäng för programstruktur.

# Betygssättning

Detaljer finns i de generella instruktionerna i INDU—Individuell Uppgift i Programmering.