```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

def task_a():
    # Read an image
    img = cv2.imread('/content/tiger.jpg')


    cv2_imshow(img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


    cv2.imwrite('output.jpg', img)
    return img

def task_b(img):
    print(f"Height: {img.shape[0]}")
    print(f"Width: {img.shape[1]}")
    print(f"Number of channels: {img.shape[2]}")
    print(f"Image size: {img.size} bytes")

def task_c(img):
    print("Blue channel (first 5x5 pixels):")
    print(img[:5, :5, 0])
    print("\nGreen channel (first 5x5 pixels):")
    print(img[:5, :5, 1])
    print("\nRed channel (first 5x5 pixels):")
    print(img[:5, :5, 2])

def task_d(img):
    # Separate channels
    b, g, r = cv2.split(img)

    # Create blank image for each channel
    zeros = np.zeros(img.shape[:2], dtype="uint8")
    blue_img = cv2.merge([b, zeros, zeros])
    green_img = cv2.merge([zeros, g, zeros])
    red_img = cv2.merge([zeros, zeros, r])


    cv2_imshow(blue_img)
    cv2_imshow(green_img)
    cv2_imshow(red_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def task_e(img):
```

```
    # Convert to grayscale
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


    cv2_imshow(gray_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    # Display shape
    print(f"Shape of grayscale image: {gray_img.shape}")

def task_f(img):
    cropped = img[100:300, 150:350]

    # Flipping
    flip_vertical = cv2.flip(img, 0)
    flip_horizontal = cv2.flip(img, 1)
    flip_both = cv2.flip(img, -1)


    cv2_imshow(cropped)
    cv2_imshow(flip_vertical)
    cv2_imshow(flip_horizontal)
    cv2_imshow(flip_both)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def task_g(img):
    # Resize width
    width_resized = cv2.resize(img, (600, img.shape[0]))

    # Resize height
    height_resized = cv2.resize(img, (img.shape[1], 400))


    cv2_imshow(width_resized) # Use cv2_imshow instead
    cv2_imshow(height_resized) # Use cv2_imshow instead
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def task_h(img):
    # Rotate 90 degrees clockwise
    rotated_90 = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)

    # Rotate 180 degrees
    rotated_180 = cv2.rotate(img, cv2.ROTATE_180)

    cv2_imshow(rotated_90) # Use cv2_imshow instead
    cv2_imshow(rotated_180) # Use cv2_imshow instead
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

```
    cv2.destroyAllWindows()

def task_i():
    # Create black image
    black_img = np.zeros((512, 512, 3), np.uint8)

    # Draw line
    cv2.line(black_img, (50, 50), (450, 50), (0, 255, 0), 5)

    # Draw rectangle
    cv2.rectangle(black_img, (50, 100), (450, 200), (255, 0, 0), 3)

    # Draw circle
    cv2.circle(black_img, (250, 350), 100, (0, 0, 255), -1)

    # Draw text
    cv2.putText(black_img, 'OpenCV', (150, 450), cv2.FONT_HERSHEY_SIMPLEX,
                2, (255, 255, 255), 3, cv2.LINE_AA)
    cv2.putText(black_img, 'Hello_People', (100, 500), cv2.FONT_HERSHEY_SIMPLEX,
                1, (255, 255, 255), 2, cv2.LINE_AA)

    # Display result
    # cv2.imshow('Drawn Shapes', black_img) # Disabled in Colab
    cv2_imshow(black_img) # Use cv2_imshow instead
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def task_j(img):

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(img, (5, 5), 0)
    edges = cv2.Canny(img, 100, 200)
    _, threshold = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)


    cv2_imshow(blurred)
    cv2_imshow(edges)
    cv2_imshow(threshold)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Main program
if __name__ == "__main__":
    # Task a
    print("Task a: Read, write and display an image")
    image = task_a()

    # Task b
    print("\nTask b: Display image properties")
    task_b(image)
```

```
# Task c
print("\nTask c: Display individual channel pixel values")
task_c(image)

# Task d
print("\nTask d: Separate color channels")
task_d(image)

# Task e
print("\nTask e: Convert to grayscale")
task_e(image)

# Task f
print("\nTask f: Cropping and flipping")
task_f(image)

# Task g
print("\nTask g: Resize width and height")
task_g(image)

# Task h
print("\nTask h: Rotate image")
task_h(image)

# Task i
print("\nTask i: Draw shapes and text")
task_i()

# Task j
print("\nTask j: Various image operations")
task_j(image)
```

```
Task a: Read, write and display an image
```

```
Task b: Display image properties
Height: 406
Width: 612
Number of channels: 3
Image size: 745416 bytes

Task c: Display individual channel pixel values
Blue channel (first 5x5 pixels):
[[82 79 73 64 58]
 [80 80 77 71 64]
 [79 80 81 77 71]
 [80 82 82 79 72]
 [83 83 81 77 71]]

Green channel (first 5x5 pixels):
[[130 127 121 112 106]
 [128 128 125 119 112]
 [127 128 129 125 119]
 [128 130 130 127 120]
 [131 131 129 125 119]]

Red channel (first 5x5 pixels):
[[164 161 155 146 140]
 [162 162 159 153 146]
 [161 162 163 159 153]
 [162 164 164 161 154]
 [165 165 163 159 153]]

Task d: Separate color channels
```

Task e: Convert to grayscale

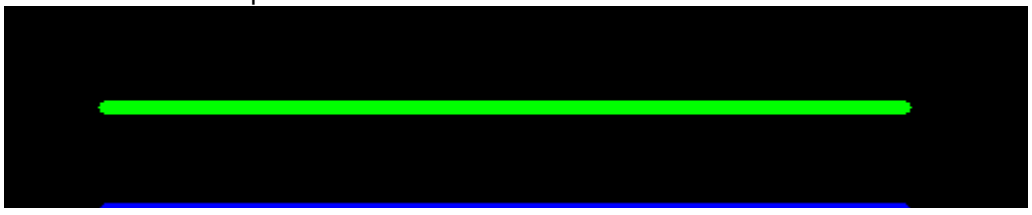
```
Shape of grayscale image: (406, 612)
```

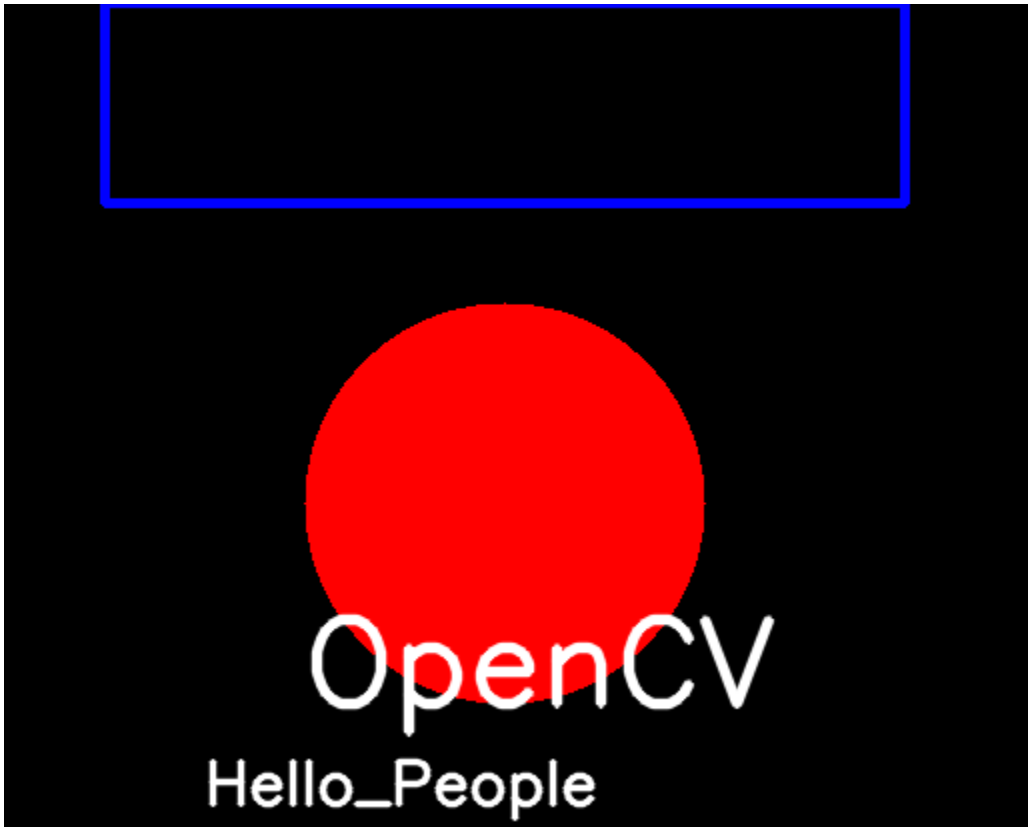
```
Task f: Cropping and flipping
```

Task g: Resize width and height

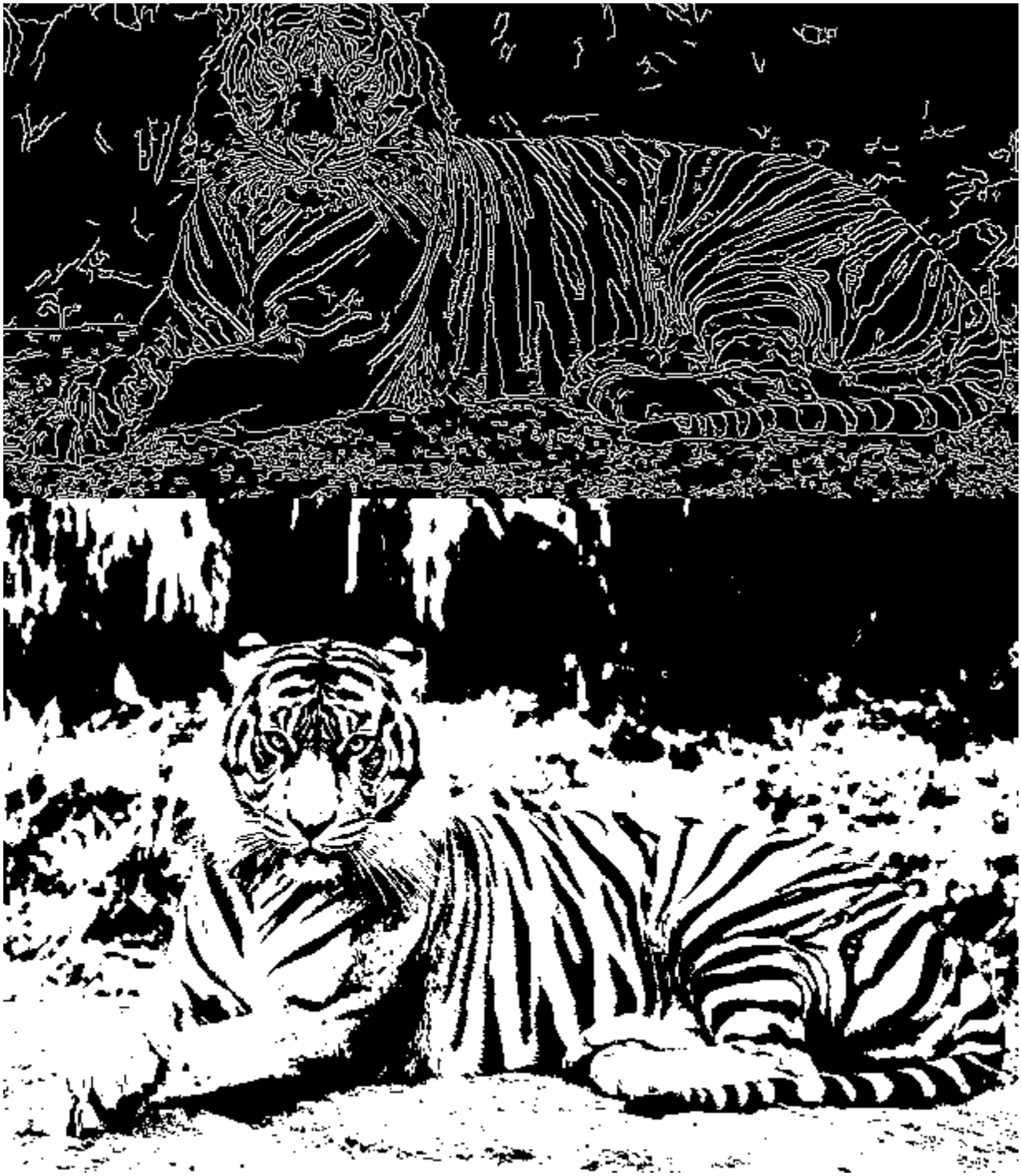
Task h: Rotate image

Task i: Draw shapes and text





Task j: Various image operations



```
!apt-get update
!apt-get install -y texlive-xetex texlive-fonts-recommended texlive-plain-generic
!pip install nbconvert
```

```
Get:1 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease [3,632 B]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64  InR
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:6 https://r2u.stat.illinois.edu/ubuntu jammy InRelease [6,555 B]
Get:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:8 https://r2u.stat.illinois.edu/ubuntu jammy/main all Packages [9,087 kB]
Get:9 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease [18.1 kB
```

```
Get:9 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease [18.1 kB
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1,26
Hit:11 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,569
Hit:13 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Get:14 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [3,103 kB
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3,415 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [4,
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [35.2
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [83.2 kB]
Get:19 https://r2u.stat.illinois.edu/ubuntu jammy/main amd64 Packages [2,758 kB]
Get:20 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy/main amd64 Packag
Fetched 26.6 MB in 4s (6,818 kB/s)
Reading package lists... Done
W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libgs9 libgs9-common libidn12
  libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1 libruby3.0
  libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13
  lmodern poppler-data preview-latex-style rake ruby ruby-net-telnet
  ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0 rubygems-integration t1utils
  teckit tex-common tex-gyre texlive-base texlive-binaries texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures tipa
  xfonts-encodings xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java
  poppler-utils ghostscript fonts-japanese-mincho | fonts-ipafont-mincho
  fonts-japanese-gothic | fonts-ipafont-gothic fonts-arphic-ukai
  fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv
  | postscript-viewer perl-tk xpdf | pdf-viewer xzdec
  texlive-fonts-recommended-doc texlive-latex-base-doc python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl
  texlive-latex-extra-doc texlive-latex-recommended-doc texlive-luatex
  texlive-pstricks dot2tex prerex texlive-pictures-doc vprerex
  default-jre-headless tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libgs9 libgs9-common libidn12
  libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1 libruby3.0
  libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13
  lmodern poppler-data preview-latex-style rake ruby ruby-net-telnet
  ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0 rubygems-integration t1utils
  teckit tex-common tex-gyre texlive-base texlive-binaries
```

```
!jupyter nbconvert --to pdf "/content/2003076_Lab1_B.ipynb"
```

```
[NbConvertApp] WARNING | pattern '/content/2003076_Lab1_B.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
```

```
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with default
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_form
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_form
--coalesce-streams
    Coalesce consecutive stdout and stderr outputs into one stream (within each cell)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_form
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.e
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter
```

```
Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.
```