```
crequire "google/cloud/vision"
# [END vision_face_detection_tutorial_imports]
# [START vision_face_detection_tutorial_process_response]
require "rmagick"
# [END vision_face_detection_tutorial_process_response]

def draw_box_around_faces path_to_image_file:, path_to_output_file:,
                          project_id:
  # [START vision_face_detection_tutorial_client]
  vision = Google::Cloud::Vision.new project: project_id
  # [END vision_face_detection_tutorial_client]

  # [START vision_face_detection_tutorial_send_request]
  image = vision.image path_to_image_file
  faces = image.faces
  # [END vision_face_detection_tutorial_send_request]

  # [START vision_face_detection_tutorial_process_response]
  image = Magick::Image.read(path_to_image_file).first

  faces.each do |face|
    puts "Face bounds:"
    face.bounds.face.each do |vector|
      puts "(#{vector.x}, #{vector.y})"
    end

    draw = Magick::Draw.new
    draw.stroke = "green"
    draw.stroke_width 5
    draw.fill_opacity 0

    x1 = face.bounds.face[0].x.to_i
    y1 = face.bounds.face[0].y.to_i
    x2 = face.bounds.face[2].x.to_i
    y2 = face.bounds.face[2].y.to_i

    draw.rectangle x1, y1, x2, y2
    draw.draw image
  end

  image.write path_to_output_file

  puts "Output file: #{path_to_output_file}"
  # [END vision_face_detection_tutorial_process_response]
end

# [START vision_face_detection_tutorial_run_application]
if __FILE__ == $PROGRAM_NAME
```

```
  project_id = ENV["GOOGLE_CLOUD_PROJECT"]

  if ARGV.size == 2
    draw_box_around_faces path_to_image_file:  ARGV.shift,
                          path_to_output_file: ARGV.shift,
                          project_id:          project_id
  else
    puts <<-usage
Usage: ruby draw_box_around_faces.rb [input-file] [output-file]

def detect_web project_id:, image_path:
  # [START vision_web_detection]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Path to local image file, eg. './image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  web = image.web

  web.entities.each do |entity|
    puts entity.description
  end

  web.full_matching_images.each do |image|
    puts image.url
  end
  # [END vision_web_detection]
end

# This method is a duplicate of the above method, but with a different
# description of the 'image_path' variable, demonstrating the gs://bucket/file
# GCS storage URI format.
def detect_web_gcs project_id:, image_path:
  # [START vision_web_detection_gcs]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Google Cloud Storage URI, eg. 'gs://my-bucket/image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  web = image.web

  web.entities.each do |entity|
    entity.description
  end

  web.full_matching_images.each do |image|
    puts image.url
  end
  # [END vision_web_detection_gcs]
end

if __FILE__ == $PROGRAM_NAME
  image_path = ARGV.shift
  project_id = ENV["GOOGLE_CLOUD_PROJECT"]

  if image_path
    detect_web image_path: image_path, project_id: project_id
  else
    puts <<-usage
Usage: ruby detect_web.rb [image file path]

def detect_text project_id:, image_path:
  # [START vision_text_detection]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Path to local image file, eg. './image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  puts image.text
  # [END vision_text_detection]
end

# This method is a duplicate of the above method, but with a different
# description of the 'image_path' variable, demonstrating the gs://bucket/file
# GCS storage URI format.
def detect_text_gcs project_id:, image_path:
  # [START vision_text_detection_gcs]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Google Cloud Storage URI, eg. 'gs://my-bucket/image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  puts image.text
  # [END vision_text_detection_gcs]
```

```
end

if __FILE__ == $PROGRAM_NAME
  image_path = ARGV.shift
  project_id = ENV["GOOGLE_CLOUD_PROJECT"]

  if image_path
    detect_text image_path: image_path, project_id: project_id
  else
    puts <<-usage
Usage: ruby detect_text.rb [image file path]

def detect_landmarks project_id:, image_path:
  # [START vision_landmark_detection]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Path to local image file, eg. './image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  image.landmarks.each do |landmark|
    puts landmark.description

    landmark.locations.each do |location|
      puts "#{location.latitude}, #{location.longitude}"
    end
  end
  # [END vision_landmark_detection]
end

# This method is a duplicate of the above method, but with a different
# description of the 'image_path' variable, demonstrating the gs://bucket/file
# GCS storage URI format.
def detect_landmarks_gcs project_id:, image_path:
  # [START vision_landmark_detection_gcs]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Google Cloud Storage URI, eg. 'gs://my-bucket/image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  image.landmarks.each do |landmark|
    puts landmark.description
    landmark.locations.each do |location|
      puts "#{location.latitude}, #{location.longitude}"
    end
  end
  # [END vision_landmark_detection_gcs]
end

if __FILE__ == $PROGRAM_NAME
  image_path = ARGV.shift
  project_id = ENV["GOOGLE_CLOUD_PROJECT"]

  if image_path
    detect_landmarks image_path: image_path, project_id: project_id
  else
    puts <<-usage
Usage: ruby detect_landmarks.rb [image file path]

def detect_image_properties project_id:, image_path:
  # [START vision_image_property_detection]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Path to local image file, eg. './image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  image.properties.colors.each do |color|
    puts "Color #{color.red}, #{color.green}, #{color.blue}"
  end
  # [END vision_image_property_detection]
end

# This method is a duplicate of the above method, but with a different
# description of the 'image_path' variable, demonstrating the gs://bucket/file
# GCS storage URI format.
def detect_image_properties_gcs project_id:, image_path:
  # [START vision_image_property_detection_gcs]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Google Cloud Storage URI, eg. 'gs://my-bucket/image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  image.properties.colors.each do |color|
    puts "Color #{color.red}, #{color.green}, #{color.blue}"
  end
  # [END vision_image_property_detection_gcs]
end

if __FILE__ == $PROGRAM_NAME
  image_path = ARGV.shift
  project_id = ENV["GOOGLE_CLOUD_PROJECT"]

  if image_path
    detect_image_properties image_path: image_path, project_id: project_id
  else
    puts <<-usage
Usage: ruby detect_image_properties.rb [image file path]

def detect_document_text project_id:, image_path:
  # [START vision_fulltext_detection]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Path to local image file, eg. './image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  document = image.document

  puts document.text
  # [END vision_fulltext_detection]
end

# This method is a duplicate of the above method, but with a different
# description of the 'image_path' variable, demonstrating the gs://bucket/file
# GCS storage URI format.
def detect_document_text_gcs project_id:, image_path:
  # [START vision_fulltext_detection_gcs]
  # project_id = "Your Google Cloud project ID"
  # image_path = "Google Cloud Storage URI, eg. 'gs://my-bucket/image.png'"

  require "google/cloud/vision"

  vision = Google::Cloud::Vision.new project: project_id
  image  = vision.image image_path

  document = image.document

  puts document.text
  # [END vision_fulltext_detection_gcs]
end
```

```
if __FILE__ == $PROGRAM_NAME
  image_path = ARGV.shift
  project_id = ENV["GOOGLE_CLOUD_PROJECT"]

  if image_path
    detect_document_text image_path: image_path, project_id: project_id
  else
    puts <<-usage
Usage: ruby detect_document_text.rb [image file path]

end
```