Article
Yuri Marx · Jan 9, 2023 7m read

Open Exchange

Generate and read QR Codes and Barcodes with Python and IRIS

Applications that work with bill payments and receipts, as well as the delivery and inventory of items, generally require the use of barcodes or QR Codes. The latter is used in even broader scenarios since the QR Code can store more information than a simple bar code. Thus, it is important to have the ability to generate barcodes and QR Codes or read the data stored in them from an image or a PDF. This article will show you how to do this using Python and some of its free libraries.

Pyzbar library

The pyzbar library reads one-dimensional barcodes and QR codes from Python 2 and 3 using the zbar library. it has these characteristics:

- Pure Python.
- Works with PIL / Pillow images, OpenCV / ImageIO / NumPy ndarrays, and raw bytes.
- Decodes locations of barcodes.
- No dependencies other than the zbar library itself.
- Tested on Python 2.7, and Python 3.5 to 3.10.

More details: https://github.com/NaturalHistoryMuseum/pyzbar/

Zbar library

ZBar Bar Code Reader is an open source software suite for reading barcodes from various sources, such as video streams, image files and raw intensity sensors. It supports EAN-13/UPC-A, UPC-E, EAN-8, Code 128, Code 93, Code 39, Codabar, Interleaved 2 of 5 and QR Code. These are the Python bindings for the library.
More details: https://sourceforge.net/p/zbar/code/ci/default/tree/python/

Python-barcode library

Python-barcode provides a simple way to create barcodes in Python. There are no external dependencies when generating SVG files. Pillow is required for generating images (e.g.: PNGs). It supports Python 3.7 to 3.10.
More details: https://github.com/WhyNotHugo/python-barcode

Pillow library

The Python Imaging Library (Pillow) adds image processing capabilities to your Python interpreter. This library provides extensive file formatting support, an efficient internal representation, and powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.
More details: https://pillow.readthedocs.io/en/stable/

Opencv-Python library

It is a wrapper in Python for OpenCV (Open-Source Computer Vision Library). It is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

Pypdfium2 library

It is an ABI-level Python 3 binding to PDFium, a powerful and liberal-licensed library for PDF creation, inspection, manipulation, and rendering. It is built with the help of ctypesgen and external PDFium binaries. Its custom setup infrastructure provides a seamless packaging and installation process. A wide range of platforms and Python versions is supported with wheel packages. It includes helper classes to simplify common use cases, while the raw PDFium/ctypes API remains accessible as well.
More details: https://pypdfium2.readthedocs.io/en/stable/

Sample Application

You can get a sample application using these Python libraries to generate and read barcodes and QR codes from images and PDFs.
Follow the steps:

1. Clone/git pull the repo into any local directory

```
$ git clone https://github.com/yurimarx/iris-qr-barcode-utils.git
```

2. Open a Docker terminal in this directory and run:

```
$ docker-compose build
```

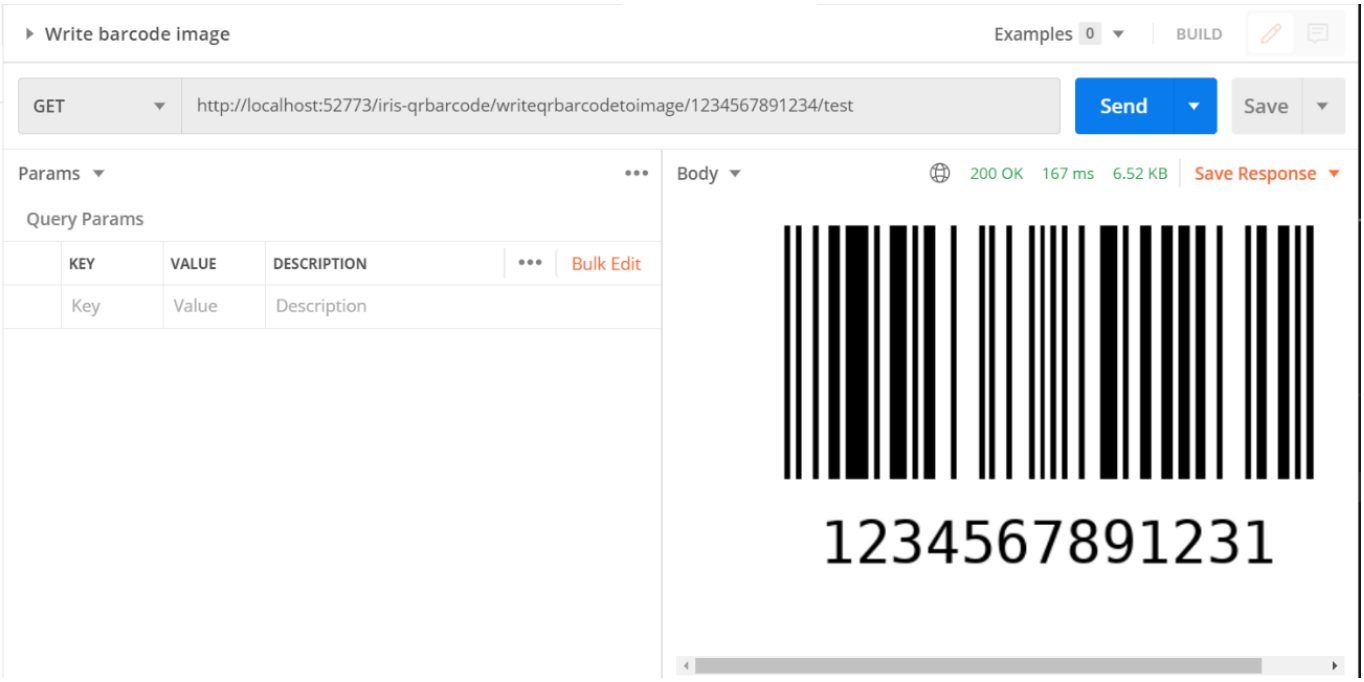
3. Run the IRIS container:

```
$ docker-compose up -d
```

4. Use Postman (or another REST Client) to write a barcode to a value.
P.S.: if you wish, you can import the Postman sample requests from: https://github.com/yurimarx/iris-qr-barcode-utils/raw/main/barcode.postm...

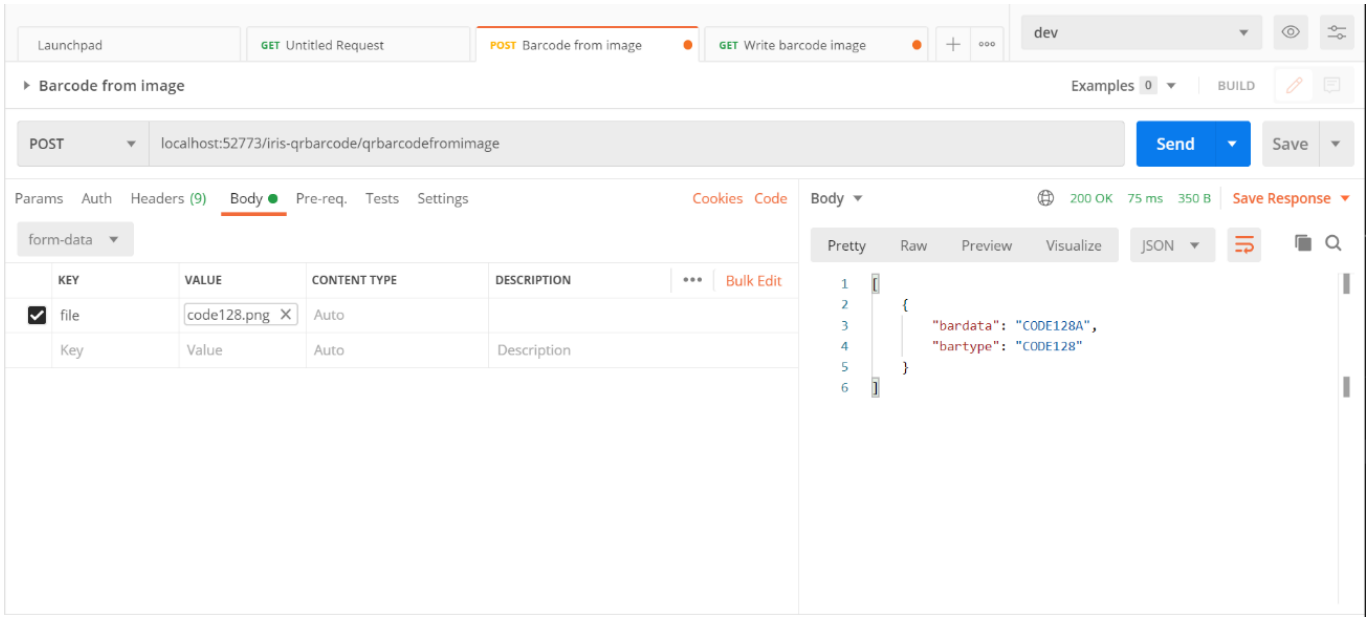
Use Postman to write a QR Code to an Image

- **Method:** GET
- **URL**: http://localhost:52773/iris-qrbarcode/writeqrbarcodetoimage/123456789123... (template is: /writeqrbarcodetoimage/barcode number/image name)



Use Postman to read a barcode value (try EAN 128 type - the project has a sample on project-folder/code128.png)

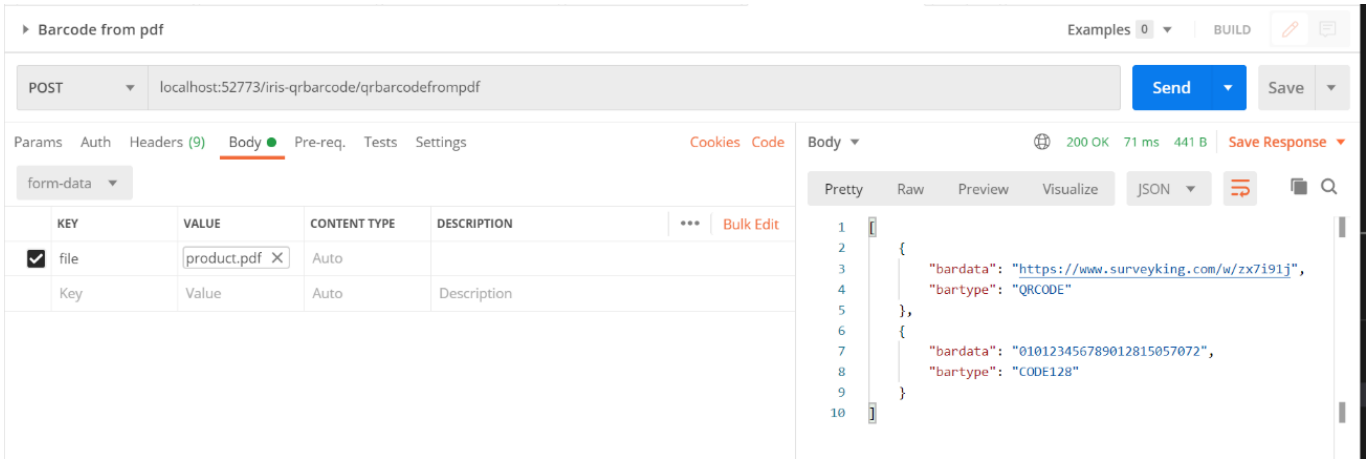
- **Method:** POST
- **URL:** http://localhost:52773/iris-qrbarcode/qrbarcodefromimage
- **Body:** form-data
- **Key:** file
- **Value:** select a file from your computer



Use Postman to read a barcode value from a PDF (try a PDF with barcode and QR Code - the project has a sample on project-folder/product.pdf)

- **Method:** POST
- **URL:** http://localhost:52773/iris-qrbarcode/qrbarcodefrompdf
- **Body:** form-data
- **Key:** file

- **Value:** select a file from your computer (sample on project-folder/product.pdf)



Source code to generate barcodes

To generate barcodes in the /iris-qrbarcode/writeqrbarcodetoimage/ API, the python-barcode library was used. This library supports the following barcode standards:

- code 39
- code 128
- PZN7 (aka PZN)
- EAN-13
- EAN-8
- JAN
- ISBN-13
- ISBN-10
- ISSN
- UPC-A
- EAN14
- GS1-128

In the dc.qrbarcode.QRBarcodeService class, in the WriteBarcodeToImage class method we have an example of how to use python-barcode to generate a barcode in the EAN13 standard (thirteen numeric positions):

```
ClassMethod WriteBarcodeToImage(number, filename) [ Language = python ]
{
    from barcode import EAN13
    from barcode.writer import ImageWriter
    import base64

    my_code = EAN13(number, writer=ImageWriter())
    image = my_code.save(filename)
    return image
}
```

The first step is to import the class that corresponds to the desired type (EAN13) from the code snippet:

```
from barcode import EAN13
```

After that just call the class constructor, then pass the number to be generated, and an ImageWriter:

```
my_code = EAN13(number, writer=ImageWriter())
```

Source code to read barcodes from images

The library we used to read barcodes and QR codes is pyzbar. This library supports EAN/UPC, Code 128, Code 39, Interleaved 2 of 5, and QR Codes. In the dc.qrbarcode.QRBarcodeService class, in the ReadQRBarcodeFromImage class method we have an example of how to utilize pyzbar to read a barcode or QR Codes:

```
ClassMethod ReadQRBarcodeFromImage(image) [ Language = python ]
{
    import cv2
    from pyzbar.pyzbar import decode
    import json

    # Make one method to decode the barcode
    class IrisBarcode:
        def __init__(self, bardata, bartype):
            self.bardata = bardata
            self.bartype = bartype


    result = []

    # read the image in numpy array using cv2
    img = cv2.imread(image)

    # Decode the barcode image
    detectedBarcodes = decode(img)

    # If not detected then print the message
    if detectedBarcodes:
        for barcodeitem in detectedBarcodes:
            item = IrisBarcode(barcodeitem.data.decode("utf-8"), barcodeitem.type)
            result.append(item)

    return json.dumps(result, default=vars)
}
```

The OpenCV (class cv2) is used to transform an image into a NumPy array. It allows pyzbar's decode method to read this array and detect barcodes. The barcodes found are returned in a JSON with the barcode type and their data.

Source code to read barcodes from PDF

The first library we used is pypdfium2. It is employed to turn each PDF page into an image and then utilize pyzbar to read the barcodes of the PDF page that has been converted into an image. In the dc.qrbarcode.QRBarcodeService class, in the ReadQRBarcodeFromPDF class method we have an example of how to employ pyzbar to read a barcode or QR Codes from PDF files:

```
ClassMethod ReadQRBarcodeFromPDF(pdfpath) [ Language = python ]
{
    from pyzbar.pyzbar import decode
    import pypdfium2 as pdfium
    import json
```

```
    class IrisBarcode:
        def __init__(self, bardata, bartype):
            self.bardata = bardata
            self.bartype = bartype


    pdf = pdfium.PdfDocument(pdfpath)
    result = []
    pages = len(pdf)

    for currentPage in range(pages):

        page = pdf.get_page(currentPage)
        pil_image = page.render_to(
            pdfium.BitmapConv.pil_image,
        )

        # Decode the barcode image
        detectedBarcodes = decode(pil_image)

        # If not detected then print the message
        if detectedBarcodes:
            for barcodeitem in detectedBarcodes:
                item = Iri
sBarcode(barcodeitem.data.decode("utf-8"), barcodeitem.type)
                result.append(item)

    return json.dumps(result, default=vars)
}
```

The pdfium.PdfDocument class is used to read each PDF page one by one and convert them into separate images in Pillow format. Then, pyzbar's decode method detects existing barcodes in the Pillow representation of the image. Finally, the barcodes and QR Codes found are returned in JSON format.

A bonus to you

If you want to generate a QR Code, it is possible to use the qrcode-library (https://pypi.org/project/qrcode/). Check out this example

```
import qrcode
img = qrcode.make('Some data here')
type(img)  # qrcode.image.pil.PilImage
img.save("some_file.png")
```

Conclusion

The use of embedded Python allows IRIS applications to read and write barcodes and QR Codes easily and for any scenario, whether in the form of a REST API or an interoperability production. It is also possible to transform data from tables into barcodes by creating procedures in Python. There are many possibilities, including those for IRIS Analytics dashboards. So enjoy!

#Embedded Python #REST API #InterSystems IRIS
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/generate-and-read-qr-codes-and-barcodes-python-and-iris