```
import cv2
import numpy as np
import scipy.fftpack
import os
import re
from PIL import Image
import PIL.ImageOps
import sys

from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///app.db'
db = SQLAlchemy(app)

class info(db.Model):
      __tablename__="info"
      number = db.Column(db.String(10), primary_key=True)
      name = db.Column(db.String(45))
      type = db.Column(db.String(20))
      rate = db.Column(db.Integer)
      amount = db.Column(db.Integer)

if len(sys.argv) != 2:
    print "Please exwcute as : python %s input_file_name \n" %
(sys.argv[0])
    sys.exit()
else:
    name = sys.argv[1]

if not os.path.isfile(name):
    print "No such file '%s'" % name
    sys.exit()

def imclearborder(imgBW, radius):

    imgBWcopy = imgBW.copy()
    contours,hierarchy = cv2.findContours(imgBWcopy.copy(),
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    imgRows = imgBW.shape[0]
    imgCols = imgBW.shape[1]
    contourList = []

    for idx in np.arange(len(contours)):
        cnt = contours[idx]

        for pt in cnt:
            rowCnt = pt[0][1]
            colCnt = pt[0][0]

            check1 = (rowCnt >= 0 and rowCnt < radius) or (rowCnt >=
imgRows-1-radius and rowCnt < imgRows)
```

```
            check2 = (colCnt >= 0 and colCnt < radius) or (colCnt >= 
imgCols-1-radius and colCnt < imgCols)

            if check1 or check2:
                contourList.append(idx)
                break

    for idx in contourList:
        cv2.drawContours(imgBWcopy, contours, idx, (0,0,0), -1)

    return imgBWcopy

def bwareaopen(imgBW, areaPixels):
    imgBWcopy = imgBW.copy()
    contours,hierarchy = cv2.findContours(imgBWcopy.copy(), 
cv2.RETR_LIST,
        cv2.CHAIN_APPROX_SIMPLE)

    for idx in np.arange(len(contours)):
        area = cv2.contourArea(contours[idx])
        if (area >= 0 and area <= areaPixels):
            cv2.drawContours(imgBWcopy, contours, idx, (0,0,0), -1)

    return imgBWcopy

filename_base = os.path.splitext(name)[0]
cmd = 'convert %s -resize 640x480\! %s.png' % (name,filename_base)
os.system(cmd)
picname = filename_base + ".png"
img = cv2.imread(picname, 0)
rows = img.shape[0]
cols = img.shape[1]
img = img[:, 59:cols-20]
rows = img.shape[0]
cols = img.shape[1]
imgLog = np.log1p(np.array(img, dtype="float") / 255)
M = 2*rows + 1
N = 2*cols + 1
sigma = 10
(X,Y) = np.meshgrid(np.linspace(0,N-1,N), np.linspace(0,M-1,M))
centerX = np.ceil(N/2)
centerY = np.ceil(M/2)
gaussianNumerator = (X - centerX)**2 + (Y - centerY)**2
Hlow = np.exp(-gaussianNumerator / (2*sigma*sigma))
Hhigh = 1 - Hlow
HlowShift = scipy.fftpack.ifftshift(Hlow.copy())
HhighShift = scipy.fftpack.ifftshift(Hhigh.copy())
If = scipy.fftpack.fft2(imgLog.copy(), (M,N))
Ioutlow = scipy.real(scipy.fftpack.ifft2(If.copy() * HlowShift, (M,N)))
Iouthigh = scipy.real(scipy.fftpack.ifft2(If.copy() * HhighShift, (M,N)))
gamma1 = 0.3
gamma2 = 1.5
Iout = gamma1*Ioutlow[0:rows,0:cols] + gamma2*Iouthigh[0:rows,0:cols]
Ihmf = np.expm1(Iout)
Ihmf = (Ihmf - np.min(Ihmf)) / (np.max(Ihmf) - np.min(Ihmf))
Ihmf2 = np.array(255*Ihmf, dtype="uint8")
Ithresh = Ihmf2 < 65
Ithresh = 255*Ithresh.astype("uint8")
Iclear = imclearborder(Ithresh, 5)
Iopen = bwareaopen(Iclear, 120)

cv2.imwrite('output.png', Iopen)
image = Image.open('output.png')
inverted_image = PIL.ImageOps.invert(image)
inverted_image.save('output.png')

cmd = 'convert output.png output.tiff'
os.system(cmd)
cmd = 'tesseract output.tiff out -l eng nobatch goodchars'
os.system(cmd)
input_file = open('out.txt')
lines = input_file.readlines()
line = " ".join([x.strip() for x in lines])
s=line.strip()
m=re.search("[A-Z][A-Z][0-9][0-9][A-Z]{1,2}[0-9]{1,4}",s)
if m:
  line=m.group(0)
  print line
  number=line
  i=info.query.get(number)
  i.amount=i.amount+i.rate
  db.session.commit()
input_file.close()
os.remove(picname)
os.remove('output.png')
os.remove('output.tiff')
os.remove('out.txt')
```