Распознавание человека в зоне просматриваемой веб-камерой, отправка уведомления на почту пользователя о произошедшем событии.

Отдельный файл с настройками почтового ящика пользователя и содержанием сообщения:

```
user = 'my_mail@gmail.com'
key = 'my_mail_password'

subject = 'Warning! Human detected.'
text = 'Human detected in your guarded zone. You can see photo on your private page.'
```

Код отвечающий за отправку сообщения при обнаружении человека на фото:

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import gmail_settings as gs

test_server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
test_receiver = 'TEST@mail.ru'

def send_email_warning(receiver=test_receiver, server=test_server, sender=gs.user,
                       password=gs.key, subject=gs.subject, message=gs.text):
    mssg = MIMEMultipart()
    mssg['From'] = sender
    mssg['To'] = receiver
    mssg['Subject'] = subject
    mssg.attach(MIMEText(message))

    server.login(sender, password)
    server.sendmail(mssg['From'], mssg['To'], mssg.as_string())
    server.quit()

send_email_warning()
```

Класс обрабатывает и преобразовывает полученное изображение, использует уже обученную модель для распознавания человека:

```
from torchvision.models import detection
import numpy as np
import torch as tr
import cv2 as cv


class HumanChecker:
    def __init__(self, device='cuda', thr=0.6):
        self.DEV = tr.device(device)
        self.THR = float(thr)
        self.model = detection.retinanet_resnet50_fpn(pretrained=True)
        self.model.eval()
        self.model = self.model.to(self.DEV)

    def preprocess_image(self, image):
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
        image = image.transpose((2, 0, 1))
        image = image / 255.0
        image = tr.FloatTensor(image)
        return image

    def __call__(self, image):
        input_image = self.preprocess_image(image)
        input_image = input_image.to(self.DEV)
        with tr.no_grad():
            result = self.model([input_image])
        boxes, scores, labels = (
            result[0]['boxes'].cpu().numpy().astype(np.int32),
            result[0]['scores'].cpu().numpy(),
            result[0]['labels'].cpu().numpy()
                                )
        detected = [(b, s, l) for b, s, l in zip(boxes, scores, labels) if s > self.THR and l == 1]
        contains_human = len(detected) > 0
        vis_image = image.copy()
        for bbox, score, label in detected:
            vis_image = cv.rectangle(vis_image, (bbox[0], bbox[1]), (bbox[2], bbox[3]),
                                     (0, 255, 0), thickness=2)
        return contains_human, vis_image

if __name__ == '__main__':
    human_checker = HumanChecker(device='cuda', thr=0.6)

    image = cv.imread('_image150.jpg')
    with_human, vis_image = human_checker(image)
    cv.imwrite('test_out.jpg', vis_image)
    print(with_human)
```