**《嵌入式智能控制技术与Python编程》**

**期末大作业**

**授课教师：孟令军**

**姓名：李泽宙**

**学号：SZ202305031**

**时间：2024年5月14日**

# 第一次作业

## 题目要求：

学习jupyter notebook的安装与基本操作方法

python编程：生成一个乘法表格，并将其写入一个txt文件中

文件名格式：按系统时间命名，例如：

20240311152039.txt

年月日时分秒.txt

注意：年的长度为4，其余均为2，数值小于10时，自动在前面补零

## 答案：

安装jupyter

python --version

3.11.8

使用pip安装jupyter

pip install jupyter

### 代码：

*import* time

current\_time = time.strftime('%Y%m%d%H%M%S', time.localtime())

file\_path = current\_time + ".txt"

*## debug*

print(file\_path)

fp = open(file\_path, "w")

*def* *main*():

*for* row *in* range(1, 10):

*for* column *in* range(1, row+1):

*#fp.write("%d \* %d = %d ", row+1, column+1, (row+1) \* (column+1))*

           result\_str = "" + str(row) + "\*" + str(column) + "=" + str(row \* column) + " "

           fp.write(result\_str)

       fp.write("\n")

   fp.close()

main()

### 执行结果：

>cat 20240317143959.txt

1\*1=1

2\*1=2 2\*2=4

3\*1=3 3\*2=6 3\*3=9

4\*1=4 4\*2=8 4\*3=12 4\*4=16

5\*1=5 5\*2=10 5\*3=15 5\*4=20 5\*5=25

6\*1=6 6\*2=12 6\*3=18 6\*4=24 6\*5=30 6\*6=36

7\*1=7 7\*2=14 7\*3=21 7\*4=28 7\*5=35 7\*6=42 7\*7=49

8\*1=8 8\*2=16 8\*3=24 8\*4=32 8\*5=40 8\*6=48 8\*7=56 8\*8=64

9\*1=9 9\*2=18 9\*3=27 9\*4=36 9\*5=45 9\*6=54 9\*7=63 9\*8=72 9\*9=81

# 第二次作业

## **题目要求**：

1. 做一些Numpy练习

2. 利用Pandas，读写CSV格式文件，包括3列：学号，姓名，成绩。共计20行。

3. 利用Pandas，读写Excel表格文件，将2中的数据，修改为100行，然后写到一个Excel表格文件中

4. 利用matplotlib，在3的Excel文件中添加一个成绩分布图、一个曲线图。

## 答案

### 代码：

*import* pandas *as* pd

*import* matplotlib.pyplot *as* plt

*import* csv

*import* random

*#from openpyxl import Workbook*

*import* openpyxl

*from* openpyxl.drawing.image *import* Image

data\_rows =[

        [100,"李老大",120],

        [101,"李老二",111],

        [102,"李老三",111],

        [103,"李老四",112],

        [104,"李老五",113],

        [105,"李老六",114],

        [106,"李老七",115],

        [107,"李老八",116],

        [108,"李老九",117],

        [109,"李老十",118],

        [110,"李十一",119],

        [111,"李十二",100],

        [112,"李十三",101],

        [113,"李十四",102],

        [114,"李十五",103],

        [115,"李十六",104],

        [116,"李十七",105],

        [117,"李十八",106],

        [118,"李十九",107],

        [119,"李二十",108],

        ]

*def* *write\_2\_csv*():

    df = pd.DataFrame(data\_rows, *columns*=["id","name","points"])

    df.to\_csv("test.csv", *index*=False)

*def* *generate\_name*():*## generate 80 rows to full excel*

    first\_name = ["大","伟","超","爱国","抗美","援朝","中北","大学","丰亮","杰"]

    last\_name = ["赵","钱","孙","李","周","吴","郑","王","刘","禄"]

    tmp\_name = random.choice(last\_name) + random.choice(first\_name)

*return* tmp\_name

*def* *write\_2\_excel*(): *# extend data to 100*

    data\_form\_file = pd.read\_csv('test.csv')

    df = pd.DataFrame(data\_form\_file)

*for* i *in* range(80):

        tmp\_row = pd.DataFrame({"id":[120 + i], "name":[generate\_name()],"points":[random.randrange(60,120)]})

        df = pd.concat( [df , tmp\_row], *ignore\_index*=True, *axis* = 0  )

    df.to\_excel('test.xlsx',*index* = False)

*def* *column\_chart*():

    df = pd.read\_excel('test.xlsx', *sheet\_name* = 'Sheet1')

*# 绘制频率直方图*

    value\_counts = df["points"].value\_counts()

    value\_counts.plot(*kind* = 'bar')

    plt.show()

*def* *line\_chart*():

    df = pd.read\_excel('test.xlsx', *sheet\_name* = "Sheet1")

    df.plot(*x* = "name", *y* = "points", *kind* = "line")

    plt.show()

*def* *gen\_pic\_save\_to\_excel*():

    df = pd.read\_excel('test.xlsx', *sheet\_name* = "Sheet1")

    fig = plt.figure()

    plt.rcParams['font.sans-serif'] = ['KaiTi']

    value\_count\_frame = df["points"].value\_counts().to\_frame(*name*="count")

    value\_count\_frame = value\_count\_frame.reset\_index()

    print(value\_count\_frame)

    plt1 = plt.subplot(2,1,1)

    plt2 = plt.subplot(2,1,2)

    plt1.bar(value\_count\_frame["points"], value\_count\_frame["count"])

    plt2.plot(df["id"],df["points"],)

*# save this fig*

    fig.savefig('./fig.png')

*# insert fig into excel*

    wb = openpyxl.load\_workbook('test.xlsx')

    ws = wb.active

    img = Image('fig.png')

    ws.add\_image(img,"D4")

    wb.save('test.xlsx')

    plt.show()

write\_2\_csv()

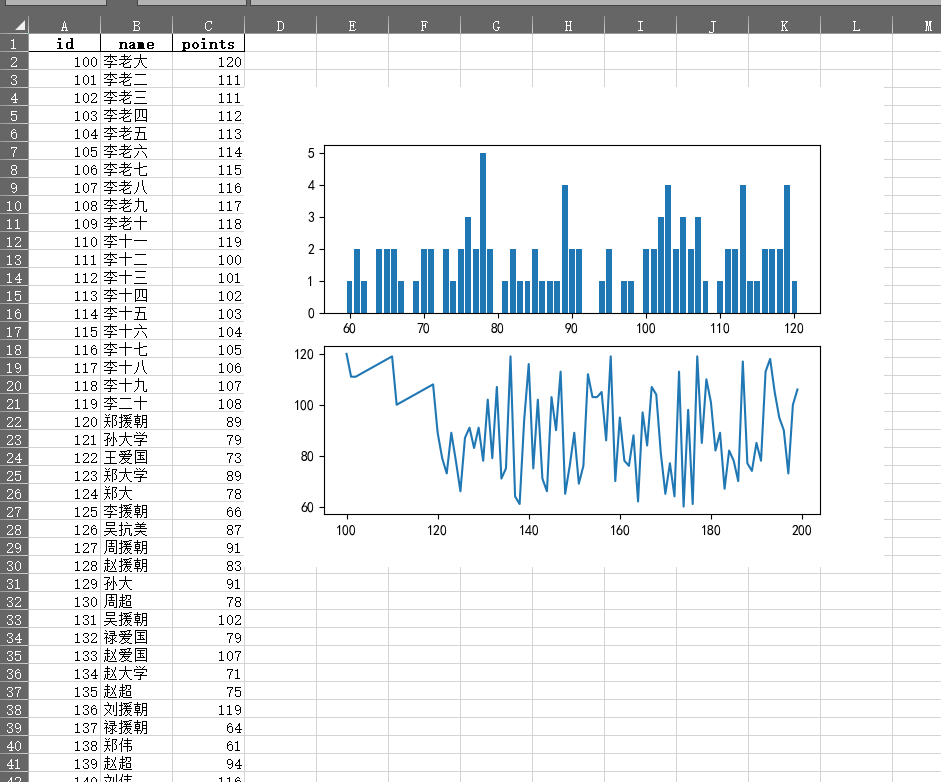
write\_2\_excel()

*#column\_chart()*

*#line\_chart()*

gen\_pic\_save\_to\_excel()

### 执行结果：



# 第三次作业

## 题目要求：

使用pillow库，结合numpy，练习图片的读写、分色显示、添加噪声、滤波、改变尺寸、旋转、多图叠加、减法等

## 答案：

### 代码：

*import* numpy *as* np

*from* PIL *import* Image

*import* matplotlib.pyplot *as* plt

*from* PIL *import* ImageFilter

*# 打开图片，旋转，然后保存旋转后的副本，用于之后的图片加减*

*def* *open\_save\_file*():

    image = Image.open("./qitingzhang.jpg")

    rotated\_image = image.rotate(180)

    small\_image = image.resize((64, 64))

    rotated\_image.save("ro\_qitingzhang.jpg")

    small\_image.save("./small\_qitingzhang.jpg")

*def* *split\_rgb*():

    image = Image.open("./qitingzhang.jpg")

    img\_r, img\_g, img\_b = image.split()

    plt.subplot(221)

    plt.axis("off")

    plt.imshow(image)

    plt.subplot(222)

    plt.axis("off")

    plt.imshow(img\_r)

    plt.subplot(223)

    plt.axis("off")

    plt.imshow(img\_g)

    plt.subplot(224)

    plt.axis("off")

    plt.imshow(img\_b)

    plt.show()

*def* *append\_noise\_and\_filter*():

*# 对图片添加噪声*

    image = Image.open("./qitingzhang.jpg")

    img\_array = np.array(image)

    h, w, c = img\_array.shape

    noise\_array = np.random.normal(0, 25, (h,w,c))

    noisy\_img\_array = np.clip(img\_array + noise\_array, 0, 255).astype(np.uint8)

    noisy\_img = Image.fromarray(noisy\_img\_array)

*# 使用pillow自带的滤波进行高斯滤波*

    after\_filter\_img = noisy\_img.filter(ImageFilter.GaussianBlur(*radius* = 2))

    plt.subplot(131)

    plt.axis("off")

    plt.imshow(image)

    plt.subplot(132)

    plt.axis("off")

    plt.imshow(noisy\_img)

    plt.subplot(133)

    plt.axis("off")

    plt.imshow(after\_filter\_img)

    plt.show()

*def* *add\_and\_sub*():

    image1 = Image.open("./qitingzhang.jpg")

    image2 = Image.open("./ro\_qitingzhang.jpg")

    image1\_array = np.array(image1)

    image2\_array = np.array(image2)

    add\_result\_array = image1\_array + image2\_array

    sub\_result\_array = np.abs(image1\_array - image2\_array)

    add\_result\_img = Image.fromarray(add\_result\_array)

    sub\_result\_img = Image.fromarray(sub\_result\_array)

    plt.subplot(121)

    plt.axis("off")

    plt.imshow(add\_result\_img)

    plt.subplot(122)

    plt.axis("off")

    plt.imshow(sub\_result\_img)

    plt.show()

open\_save\_file()

split\_rgb()

append\_noise\_and\_filter()

add\_and\_sub()

### 执行结果：

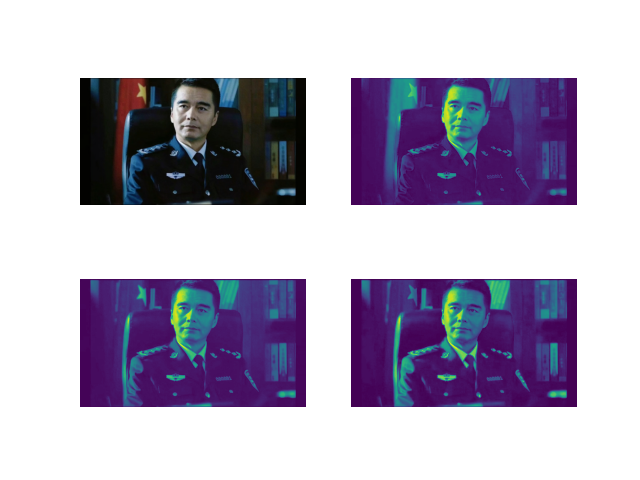


Figure 1分色

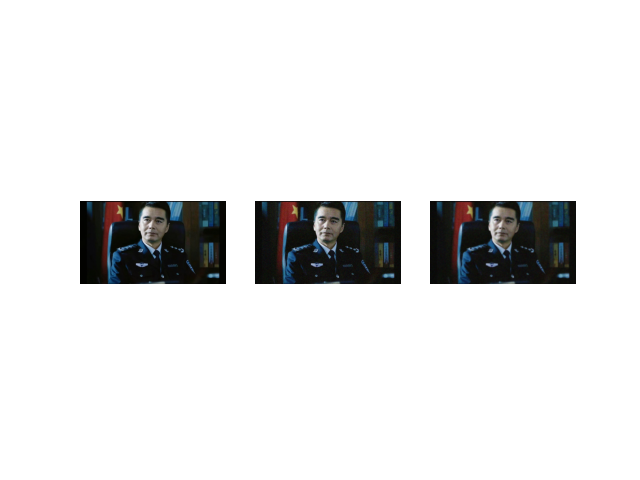


Figure 2 加噪与滤波



Figure 3 旋转



Figure 4图片修改尺寸

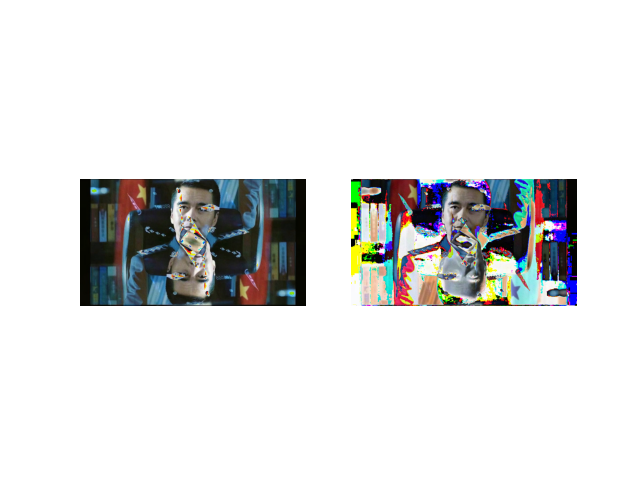


Figure 5 图片相加与相减

# 第四次作业

## 题目要求

1. 安装opencv，并练习使用opencv读写图片文件、图片的基本处理操作

2. 利用opencv打开摄像头，进行录像，在视频的左上角，添加时间码文字，格式：YY/MM/DD/HH/MM/SS.视频保存的文件名为：时间码.AVI（或者MP4）

3. 使用opencv打开录制的文件，进行回放，按‘s’键保存当前画面，文件名为：时间码.png，按'q'键，退出。

## 答案

### 代码

*import* cv2

*import* time

*import* threading

*# 读取和保存图片*

*def* *read\_and\_save\_pic*():

    img = cv2.imread("./read\_test.png")

    resized\_img = cv2.resize(img,(100,200), *interpolation*=cv2.INTER\_AREA)

    cv2.imwrite("./resized\_read\_test.png", resized\_img)

*# 录像 保存视频 在视频的左上角添加时间 yy/mm/dd/hh/mm/ss*

*def* *record\_and\_save*():

*# pass*

*# generate time stamp as file name*

    current\_time = time.strftime('%Y%m%d%H%M%S', time.localtime())

    video\_path = current\_time + ".mp4"

    print("output file name:{}".format(video\_path))

*# init cap*

    FPS = 24

    cap = cv2.VideoCapture(0)

    fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')

    wid = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

    hig = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

    print("shape of video {} \* {}".format(wid, hig))

    vid\_out = cv2.VideoWriter(video\_path, fourcc, FPS, (wid, hig) )

*while*(cap.isOpened):

        ret, frame = cap.read()

*# if ret:*

        font = cv2.FONT\_HERSHEY\_SIMPLEX

        current\_time = time.strftime('%Y%m%d%H%M%S', time.localtime())

        frame = cv2.putText(frame, current\_time, (10,50), font, 1, (255,255,255), 2, cv2.LINE\_AA)

        vid\_out.write(frame)

        cv2.imshow("frame",frame)

*if* cv2.waitKey(1) *==* ord('q'):

*break*

    cap.release()

    vid\_out.release()

    cv2.destroyAllWindows()

*# 打开录制的视频，回放，按s保存当前画面，保存为 时间码.png 按q退出*

*def* *playback\_save\_pic*():

    video\_path = "./20240411152831.mp4"

    cap = cv2.VideoCapture(video\_path)

*while*(cap.isOpened):

        ret, frame = cap.read()

        cv2.imshow("frame", frame)

*if* cv2.waitKey(1) *==* ord('s'):

            current\_time = time.strftime('%Y%m%d%H%M%S', time.localtime())

            img\_path = current\_time + ".png"

            cv2.imwrite(img\_path, frame)

    cap.release()

    cv2.destroyAllWindows()

*## 经测试可用*

*# read\_and\_save\_pic()*

*# record\_and\_save()*

*# playback\_save\_pic()*

### 执行结果：

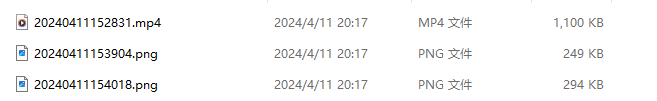


Figure 6 捕获的视频



Figure 7 视频截图

# 第五次作业

## 题目要求

利用opencv及相关的库，编程实现一个车牌照片中的车牌识别

给出算法步骤和流程图

网络调研我国的标准车牌类型

针对不同颜色类型的车牌，车牌识别算法方面需要作哪些改进？

识别准确率和什么因素有关？

## 答案：

### 代码

*import* cv2

*import* numpy *as* np

*import* pytesseract

*from* PIL *import* Image

*def* *show\_image*(*desc*, *image*):

    cv2.imshow(desc, image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

*def* *reg\_area\_color*(*image*):

*#找到原图像最多的颜色，当该颜色为红色或蓝色时返回该颜色的名称*

    kernel = np.ones((35, 35), np.uint8)

    hsv = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

*# 以上为图像处理*

    Open = cv2.morphologyEx(hsv, cv2.MORPH\_OPEN, kernel)

*# 对Open图像的H通道进行直方图统计*

    hist = cv2.calcHist([Open], [0], None, [180], [0, 180])

*# 找到直方图hist中列方向最大的点hist\_max*

    hist\_max = np.where(hist *==* np.max(hist))

*# hist\_max[0]为hist\_max的行方向的值，即H的值，H在0~10为红色*

*if* 0 *<* hist\_max[0] *<* 10:

        res\_color = 'red'

*elif* 100 *<* hist\_max[0] *<* 124:  *# H在100~124为蓝色*

        res\_color = 'blue'

*else*:

*# H不在前两者之间跳出函数*

        res\_color = 'unknow'

*return* res\_color

img = cv2.imread('./ori\_pic.jpeg')

*# 调整图片大小*

img = cv2.resize(img, (1024, 768))

*# 在进行识别之前，应当首先确定车牌在一张图片中的范围*

*# 1. 将图片变换到灰度空间，剔除颜色的影响*

*# 2. 在没有颜色影响后，滤波就会变得比较简单*

*# 3. 进行边缘检测*

*# 灰度图*

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

*# show\_image('灰度', gray)*

*# 双边滤波*

*# 双边滤波的好处在于，尽管自然图片临近像素点之间的变化比较平滑，但是仍要考虑到自然图片中，物体的边缘*

*# 边缘的变化仍旧是相对较大的，如果认为边缘也是平滑的话，滤波得到的结果就是边缘被模糊*

*# 双边滤波在边缘的权重有明显的分界*

blf = cv2.bilateralFilter(gray, 13, 15, 15)

*# show\_image('双边', blf)*

*# 边缘检测*

edged = cv2.Canny(blf, 30, 200)

*# show\_image('边缘', edged)*

*# 寻找轮廓（图像矩阵，输出模式，近似方法）*

contours, \_ = cv2.findContours(edged.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

*# 根据区域大小排序取前十*

contours = sorted(contours, *key*=cv2.contourArea, *reverse*=True)[:10]

screenCnt = None

*# 遍历轮廓，找到车牌轮廓*

*for* c *in* contours:

*if* cv2.contourArea(c) *>* 1024 \* 768 \* 0.05:

*continue*

*# 计算轮廓周长（轮廓，是否闭合）*

    peri = cv2.arcLength(c, True)

*# 折线化（轮廓，阈值（越小越接近曲线），是否闭合）返回折线顶点坐标*

    approx = cv2.approxPolyDP(c, 0.018 \* peri, True)

*# 获取四个顶点（即四边形, 左下/右下/右上/左上*

*if* len(approx) *==* 4:

*# [参数]左上角纵坐标:左下角纵坐标,左上角横坐标:右上角横坐标*

        crop\_image = img[approx[3][0][1]:approx[0][0][1], approx[3][0][0]:approx[2][0][0]]

*# show\_image('crop', crop\_image)*

*if* 'blue' *==* reg\_area\_color(crop\_image):

            screenCnt = approx

*break*

*# 如果找到了四边形*

*if* screenCnt *is* *not* None:

*# 根据四个顶点坐标对img画线(图像矩阵，轮廓坐标集，轮廓索引，颜色，线条粗细)*

    cv2.drawContours(img, [screenCnt], -1, (0, 0, 255), 3)

*# show\_image('contour', img)*

"""遮罩"""

*# 创建一个灰度图一样大小的图像矩阵*

mask = np.zeros(gray.shape, np.uint8)

*# 将创建的图像矩阵的车牌区域画成白色*

cv2.drawContours(mask, [screenCnt], 0, 255, -1, )

*# 图像位运算进行遮罩*

mask\_image = cv2.bitwise\_and(img, img, *mask*=mask)

*# show\_image('mask\_image', mask\_image)*

"""图像剪裁"""

*# 获取车牌区域的所有坐标点*

(x, y) = np.where(mask *==* 255)

*# 获取底部顶点坐标*

(topx, topy) = (np.min(x), np.min(y))

*# 获取底部坐标*

(bottomx, bottomy,) = (np.max(x), np.max(y))

*# 剪裁*

cropped = blf[topx:bottomx, topy:bottomy]

*# show\_image('cropped', cropped)*

"""OCR识别"""

*# print(pytesseract.get\_languages(config=''))*

text = pytesseract.image\_to\_string(Image.fromarray(cropped))

print("result " + text)

### 执行结果：



Figure 8 原始图片



Figure 9 处理后的框选结果

#### # 给出算法步骤和流程图

1. 读取图片
2. 对图片进行降噪处理
3. 寻找图片中的轮廓
4. 对轮廓面积进行排序，寻找较大的轮廓（因为图片中可能会出现很多小轮廓）
5. 将轮廓进行折线化
6. 寻找四边形的轮廓
7. 根据轮廓对原图进行反向裁切
8. 对裁切的结果图片进行OCR

#### # 网络调研我国的标准车牌类型

1. 蓝牌车 蓝底白字
2. 黄牌车 黄底黑字
3. 黑牌车 黑底白字
4. 绿牌车 绿地白字

#### # 针对不同颜色类型的车牌，车牌识别算法方面需要作哪些改进？

针对车牌颜色的不同，在识别车牌的时候，需要对车牌进行颜色的判断，以预先判断识别到的目标是否是车牌，同时，由于交通标志也存在大量的黄底黑字，蓝底白字标志，且他们大多也是四边形的，因此需要对识别到的内容做判断，以确定识别到的是包含车牌号的有效车牌

#### # 识别准确率和什么因素有关？

1. 原始图片质量
2. 原始图片的复杂程度（如果原始图片只有车牌，那么识别率肯定会大大提高）
3. 去噪算法效果
4. OCR效果

# 第六次作业

## 题目要求

1. 通过网络收发一个文件，文件类型任意。能显示发送和接收的进度。网络可以是wifi或者有线
2. 通过request获取网页，并解析出网络时间，进行终端显示

## 答案：

### 代码

文件的收发：

Client代码：

import socket

client = socket.socket()

client.connect(('localhost', 12345))

file = open('./send.txt', 'rb')

data = file.read(1024)

client.send(data)

file.close()

client.close()

server代码：

import socket

server = socket.socket()

server.bind(('localhost',12345))

server.listen(5)

print("listening...")

client, addr = server.accept()

print("connection from ", addr)

file = open("./received.txt", "wb")

data = client.recv(1024)

file.write(data)

file.close()

client.close()

server.close()

从request中解析时间

import requests

import datetime

# 发送HTTP请求

url = "http://www.baidu.com"

response = requests.get(url)

if response.status\_code == 200:

    server\_date = datetime.datetime.strptime(response.headers['Date'], '%a, %d %b %Y %H:%M:%S %Z')

    server\_date = server\_date.replace(tzinfo=datetime.timezone.utc)

    print(f"network time: {server\_date}")

else:

    print("Failed to retrieve server time")

### 执行结果：

