Level 1.1【数学】方程求解：基于Python，给定函数形式y=-2x+1，给定输  
入y，求解x

代码思路：

1. 定义x，给y赋值
2. 给出方程-2x+1-y=0
3. 用sympy库中solve解出x

代码

*from* sympy *import* \*  *#导入sympy库，用于解函数*

y=float(input("请输入一个数:"))  *#给y赋值*

x= symbols('x')    *#定义x*

print(solve(x\*\*2-2\*x+1-y,x))  *#求解x，并打印*

示例





Level 2.2【数学】微分方程计算：使用numpy、sympy、scipy、math库等工  
具，根据一阶微分方程dy/dx=y，画出自变量x在定义域(0.5)中对应的y  
值的曲线（提示：该函数为y=a\*exp(x)，a为常数）

代码思路：

1. 定义x, y，令y=f(x)
2. 给出方程 dy/dx-y=0
3. 用sympy库中dslove解出y=f(x)
4. 给a赋值(为了方便绘图)
5. 绘图

代码

*from* sympy *import* \*  *#导入sympy库，用于解函数*

*#导入绘图所需模块*

*import* numpy *as* np

*from* matplotlib *import* pyplot *as* plt

plt.rcParams['font.sans-serif'] = ['SimHei']  *#用来正常显示中文标签*

plt.rcParams['axes.unicode\_minus'] = False    *#用来正常显示负号*

f = symbols('f', *cls*=Function)                *#定义函数f*

x = symbols('x')                              *#定义x*

eq = Eq(f(x).diff(x)-f(x), 0)                 *#给出方程*

pprint(dsolve(eq, f(x)))                      *#求解并打印函数f*

c=float(input("请确定常数C1:"))                *#确定函数f中常数C1*

*#绘图*

x = np.arange(0, 5, 0.1)                      *#生成等差数组*

y = np.exp(x)\*c

plt.xlabel('x')

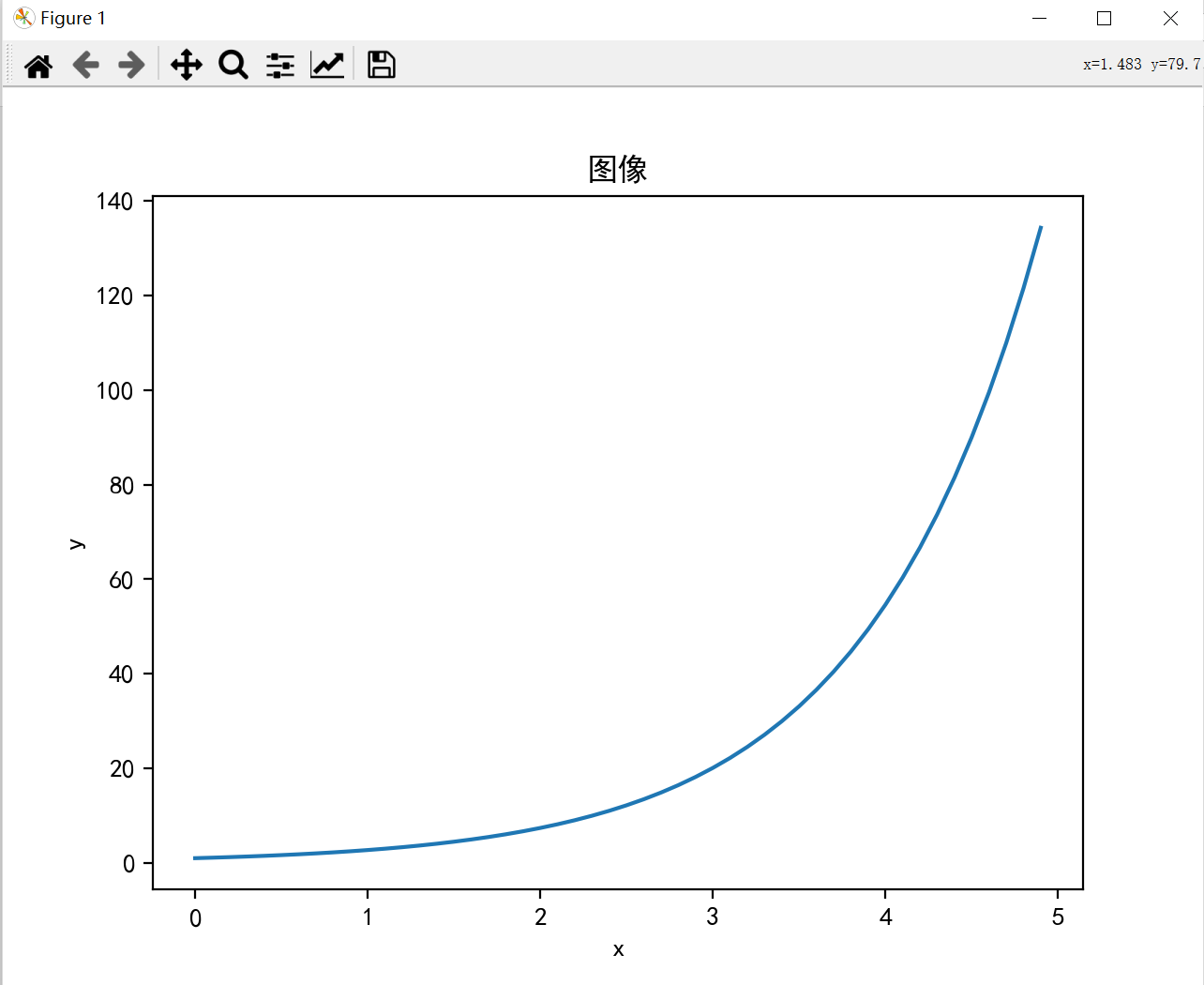
plt.ylabel('y')

plt.title("图像")

plt.plot(x, y)

plt.show()

示例



Level 3.2 【数学】微分计算：使用numpy、sympy、scipy、math库等工具求解  
二阶常系数齐次微分方程 a\*f(x) +b\*f‘(x)+c\*f”(x)=0

代码思路：

1. 定义x, f(x)
2. 给出方程a\*f(x) +b\*f‘(x)+c\*f”(x)=0
3. 给a，b，c赋值
4. 用sympy库中dslove解出y=f(x)

代码

*from* sympy *import* \*             *#导入sympy库，用于解函数*

a=int(input("请输入一个整数a:"))  *#确定方程中C1*

b=int(input("请输入一个整数b:"))  *#确定方程中C2*

c=int(input("请输入一个整数c:"))  *#确定方程中C3*

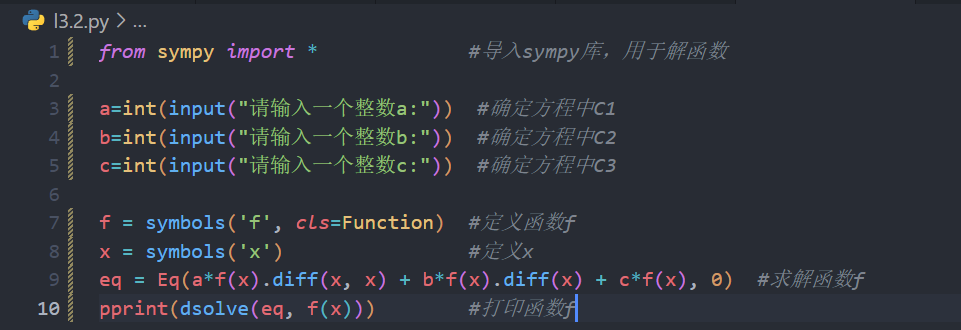
f = symbols('f', *cls*=Function)  *#定义函数f*

x = symbols('x')                *#定义x*

eq = Eq(a\*f(x).diff(x, x) + b\*f(x).diff(x) + c\*f(x), 0)  *#求解函数f*

pprint(dsolve(eq, f(x)))        *#打印函数f*

示例





L4: https://github.com/ageitgey/face\_recognition  
人脸识别：  
本项目是世界上最简洁的人脸识别库，你可以使用Python和命令行工具提取、识别、操作人脸。使用世界上最简单的面部识别库从Python或命令行识别和操作面部。使用dlib的最先进的面部识别技术和深度学习技术构建而成。该模型在 Wild 基准数据集中的Labeled Faces 上的准确性为99.38%。

代码思路：

1. 提供示例人脸图片并学会识别
2. 提供已知人脸信息
3. 用摄像头获取实时图片
4. 以 1/4 分辨率处理每个视频帧（但仍以全分辨率显示）检测每隔一帧视频中的人脸
5. 识别并显示结果，并圈出人脸，标出相关信息
6. 结束后释放网络摄像头

代码

*import* face\_recognition

*import* cv2

*import* numpy *as* np

*#为提高效率做出：*

*#   1.以 1/4 分辨率处理每个视频帧（但仍以全分辨率显示）*

*#   2.仅检测每隔一帧视频中的人脸。*

video\_capture = cv2.VideoCapture(0)  *#开摄像头*

*#加载示例图片并学习如何识别它*

jyh\_image = face\_recognition.load\_image\_file("./image/1.jpg")

jyh\_face\_encoding = face\_recognition.face\_encodings(jyh\_image)[0]

zjc\_image = face\_recognition.load\_image\_file("./image/2.jpg")

zjc\_face\_encoding = face\_recognition.face\_encodings(zjc\_image)[0]

lxy\_image = face\_recognition.load\_image\_file("./image/3.jpg")

lxy\_face\_encoding = face\_recognition.face\_encodings(lxy\_image)[0]

wxh\_image = face\_recognition.load\_image\_file("./image/4.jpg")

wxh\_face\_encoding = face\_recognition.face\_encodings(wxh\_image)[0]

zql\_image = face\_recognition.load\_image\_file("./image/4.jpg")

zql\_face\_encoding = face\_recognition.face\_encodings(zql\_image)[0]

*# 创建已知人脸编码及其名称的数组*

known\_face\_encodings = [

    jyh\_face\_encoding,

    zjc\_face\_encoding,

    lxy\_face\_encoding,

    wxh\_face\_encoding,

    zql\_face\_encoding

]

known\_face\_names = [

    "jiang yuhang",

    "zang jingcheng",

    "lu xingyu",

    "wang xihe",

    "zhang qilin"

]

*# 初始化一些变量*

face\_locations = []

face\_encodings = []

face\_names = []

process\_this\_frame = True

*while* True:

*# 抓取单帧视频*

    ret, frame = video\_capture.read()

*# 将视频帧大小调整为 1/4 大小*

    small\_frame = cv2.resize(frame, (0, 0), *fx*=0.25, *fy*=0.25)

*# 将图像从 BGR 颜色（OpenCV 使用）转换为 RGB 颜色（face\_recognition使用）*

    rgb\_small\_frame = small\_frame[:, :, ::-1]

*#仅处理每隔一帧视频以节省时间*

*if* process\_this\_frame:

*# 查找当前视频帧中的所有人脸和人脸编码*

        face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

        face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)

        face\_names = []

*for* face\_encoding *in* face\_encodings:

*#查看人脸是否与已知人脸匹配*

            matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

            name = "Unknown"

*# 如果在known\_face\_encodings中找到匹配项，只需使用第一个匹配项即可。*

*# 如果匹配项中为 True：*

*# first\_match\_index = matches.index（True）*

*# 名称 = known\_face\_names[first\_match\_index]*

*# 或者，使用与新面距离最小的已知人脸*

            face\_distances = face\_recognition.face\_distance(known\_face\_encodings, face\_encoding)

            best\_match\_index = np.argmin(face\_distances)

*if* matches[best\_match\_index]:

                name = known\_face\_names[best\_match\_index]

            face\_names.append(name)

    process\_this\_frame = *not* process\_this\_frame

*# 显示结果*

*for* (top, right, bottom, left), name *in* zip(face\_locations, face\_names):

*# 将视频帧大小调回原来大小*

        top \*= 4

        right \*= 4

        bottom \*= 4

        left \*= 4

*# 在脸部周围画一个框*

        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

*# 在脸部下方标出名字*

        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

        font = cv2.FONT\_HERSHEY\_DUPLEX

        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

*# 显示生成的图像*

    cv2.imshow('Video', frame)

*# 点击键盘上的“q”退出*

*if* cv2.waitKey(1) & 0xFF == ord('q'):

*break*

*# 释放网络摄像头*

video\_capture.release()

cv2.destroyAllWindows()

示例

代码结果实时显示，不便展示示例