# 一、Numpy 基本操作（提交所有代码截图及运行结果）

1、导入 numpy 库

2、建立一个一维数组 a，初始化为[4,5,6]

（1）输出 a 的类型（type）

（2）输出 a 的各维度大小（shape）

（3）输出 a 的第一个元素

3、建立一个二维数组 b，初始化为[[4,5,6], [1,2,3]]

（1）输出 b 的各维度大小（shape）

（2）输出b[0,0], b[0,1], b[1,1]这三个元素

4、建立矩阵

（1）建立一个大小为3 × 3的全 0 矩阵 c

（2）建立一个大小为4 × 5的全 1 矩阵 d

（3）建立一个大小为4 × 4的单位矩阵 e

5、建立一个数组 f，初始化为[0,1,2,3,4,5,6,7,8,9,10,11]（arange）

（1）输出 f 以及 f 的各维度大小

（2）将 f 的 shape 改为3 × 4（reshape）

（3）输出 f 以及 f 的各维度大小

（4）输出 f 第二行（f[1, ∶]）

（5）输出 f 最后两列（f[: ,2: ]）

（6）输出 f 第三行最后一个元素（使用-1 表示最后一个元素）

# 二、熟悉深度学习框架Pytorch/Tensorflow（提交每个练习的实现步骤描述以及下面要求提交的结果）

1、线性回归

(1) 生成训练数据，数据为带有服从-0.5 到 0.5 的均匀分布噪声的正弦函数，代码如下：

num\_observations = 100

    x = np.linspace(-3,3,num\_observations)

    y = np.sin(x) + np.random.uniform(-0.5,0.5,num\_observations)

画出这 100 个样本的散点图。**（提交散点图）**

1. 使用 Pytorch/Tensorflow 实现线性回归模型，训练参数w和b。

𝑦 = 𝑤 ∗ 𝑥 + 𝑏

(3) 输出参数 w、b 和 损失。**（提交运行结果）**

(4) 画出预测回归曲线以及训练数据散点图，对比回归曲线和散点图并分析原因。

# （提交图片及分析）

2、线性回归（使用多项式函数对原始数据进行变换）

(1) 生成训练数据，数据同上

(2) 使用 Pytorch/Tensorflow 实现线性回归模型，这里我们假设y是x的 3 次多项式，那么我们可以将数据扩展为：x、x2、x3三维数据，此时模型变为：

𝑦 = 𝑤1 ∗ 𝑥 + 𝑤2 ∗ 𝑥2 + 𝑤3 ∗ 𝑥3 + 𝑏

(3) 训练模型并输出参数w1、w2、w3、b 和 损失。**（提交运行结果）**

(4) 画出预测回归曲线以及训练数据散点图，对比回归曲线和散点图并分析原因。

# （提交图片及分析）

3、Softmax 分类

(1) 获取 MNIST 数据集，每张图片像素为28 × 28

(2) 模型架构为：

𝑦 = 𝑠𝑜𝑓𝑡𝑚𝑎𝑥(𝑤 ∗ 𝑥 + 𝑏)

其中𝑤的维度为784 × 10，𝑏的维度为10。

(3) 画出训练和测试过程的准确率随迭代次数变化图，画出训练和测试过程的损失随迭代次数变化图。**（提交最终分类精度、分类损失以及两张变化图）**