一、Python基础语法

1.简述 Python 中 变量、函数、类 的命名规则并给出每类命名的 两个示例，简要说明使用规范命名的好处。

变量：

- 命名规则：全部小写字母，单词间用下划线分隔
- 示例：user_name,total_count

函数：

- 命名规则：同变量，保持小写加下划线
- 示例：calculate_tax,send_message

类：

- 命名规则：每个单词首字母大写，不使用下划线
- 示例：MyClass,HttpRequest

规范命名的好处：

1.提升可读性：看到名字就能大致猜出用途，减少注释依赖。 2.统一风格：团队协作时代码看起来象一个人写的，降低维护成本。

2.请学习python基本的循环语句、分支语句、函数定义与调用等内容，自行完成下述两个函数：

1.函数 get_fibonacci(n)：输入整数 n，返回前 n 个斐波那契数。调用 get_fibonacci(10) 并提交输出结果。 2.函数get_primes(n)：输入整数 n，返回前 n 个质数。调用 get_primes(10) 并提交输出结果。

3.请列举 Python 中常见的数据容器，并简述每种容器的特点和其常见操作函数

列表

- 特点：列表是有序，可变的容器，可以容纳不同类型的元素，并且允许重复
- 常见操作函数：
  append(),extend(),insert(),remove(),pop(),clear(),index(),count(),reverse()sort(),len(),copy().

元组

- 特点：元组是有序的，不可变的容器，一旦创建，其元素不可修改，允许重复。
- 常见操作函数： index(),count(),len().

字典

- 特点：无序，可变，键值对映射，键必须不可变
- 常见操作函数： update(),pop(),popitem(),clear(),get(),keys(),values(),items(),len(),copy().

集合

- 特点：集合是无序、不重复的容器，支持数学上的集合运算。
- 常见操作函数：
  add(),update(),remove(),discard(),pop(),clear(),union(),intersection(),difference(),symmetric_difference(),len(),copy().

字符串

- 特点：不可变性，有序性
- 常见操作函数： len(),upper(),lower(),replace().

4.编程实例：成绩管理

- 用合适的数据结构储存这份数据。
- 计算每名学生的总分和平均分，并输出如下格式：

> 张三: 总分=255, 平均分=85.0

- 在储存的数据结构中补充遗漏数据：姓名：郑十，语文：92，数学：91，英语：86
- 找出总分最高的学生，输出姓名和总分。
- 给赵六的数学成绩加 5 分，同时更新总分和平均分，并输出修改后的信息。
- 输出每科平均分和每科最高分的学生（同分则全部列出）。

二、面向对象编程

1.请简述什么是 类（Class）、什么是 对象（Object），二者之间的关系是什么？

类

类是模板，它把一类事物的共性（属性和行为）抽象出来，形成统一的定义；

对象

对象是实例，它是类的具体化，拥有独立的状态和可执行的行为。

二者关系

- 类是对象的抽象，对象是类的具体存在。
- 一个类可以创建多个对象，每个对象都拥有类中定义的属性和方法，但各自的数据可以不同。
- 创建对象的过程叫实例化，实例化后的对象具备独立的身份和状态，可直接被程序使用。

2.在学习了类的方法和属性之后，完成下述编程实例：

1.定义一个 Student 类，要求如下：

- 属性： name（姓名） age（年龄） __scores（私有属性，包含语文、数学、英语成绩的字典）
- 方法 introduce()：打印语句如"我是张三，今年18岁，成绩：{'语文':90, '数学':85, '英语':92}"
  get_average()：返回平均成绩 update_score(subject, value)：修改某一科成绩

2.创建 2 个学生对象，分别调用 introduce() 和 get_average()，输出结果。
3.测试封装效果： 尝试直接修改私有属性，例如：s1.__scores['数学'] = 100，观察会发生什么，并说明原因 使用 s1.update_score("数学", 100) 正确更新成绩，并调用 get_average() 查看结果

3.请解释什么是继承，它的好处是什么？在第二题基础上完成下述编程实例：

1.定义一个 GraduateStudent 类，继承自 Student，新增：

- 属性：research_topic（研究方向）
- 方法： 1.introduce() 重写，输出内容包含研究方向。例如："我是李四，今年24岁，研究方向：机器学习" 2.请创建一个研究生对象，调用 introduce() 方法，验证继承和方法重写。

> 继承 1.定义 继承是面向对象编程里"子类"可以自动获得"父类"已有属性和方法的一种机制。 2.好处 代码复用：不用从头写重复的代码，子类直接沿用父类的功能 易维护：父类修改后，所有子类自动生效，一处改，处处改 可扩展：在不改动原有类的前提下，子类可以新增属性或方法，随时扩展功能。 接口一致：所有子类都遵循父类的接口规范，便于统一调用和管理。

4.请解释什么是多态，并完成下述编程实例：

1.定义两个类：

- Dog，有方法 speak()，输出 "汪汪"
- Cat，有方法 speak()，输出 "喵喵"

2.编写一个函数 animal_speak(animal)，传入任意动物对象，调用其 speak() 方法并打印结果。 3.创建一个 Dog 对象和一个 Cat 对象，分别调用 animal_speak()，查看输出。

> 多态是面向对象编程的核心特性之一，它允许同一操作作用于不同类型的对象时，自动选择对应的实现方式，从而产生不同的行为或结果。简单来说就是"一个接口，多种实现"。

5.什么是迭代器（Iterator）？它和可迭代对象（Iterable）有什么区别？Python 中如何使用 iter() 和 next()？请写一个简单示例，遍历列表 [1, 2, 3, 4] 并输出每个元素。

> 一.迭代器 迭代器是一个"能记住遍历位置"的对象，它实现了两个核心方法： iter_():返回自身 next_():返回下一个元素，遍历完毕后抛出StopIteration异常 迭代器只能向前遍历，且"一次性消费"——遍历完即耗尽，需重新创建才能再次使用。 二.区别 1.迭代器实际执行遍历，只能遍历一次；可迭代容器提供"容器"或"生成规则"，可被多次遍历。 2.迭代器必须同时实现_iter_()和_next_();可迭代对象只需_iter()_(返回迭代器)。3.迭代器的内存占用极低（按需生成，惰性计算）；可迭代对象的内存占用可能较高（如列表存储全部数据）。 4.迭代器的类型有由iter()返回的列表迭代器、自定义迭代器类等；可迭代对象的常见类型有列表、字符串、字典、文件对象、生成器等。

6.编程实例：Mini-Batch 数据生成器

背景：在机器学习训练中，我们通常不会一次性把整个数据集喂给模型，而是将数据分成若干小批次（mini-batch）迭代训练。 题目要求：请你实现一个类 BatchDataLoader，它可以将给定数据集分批次返回，每次迭代返回一个 batch。功能要求：

- 类初始化： data：一个列表，包含所有样本，例如 [1,2,3,...,20] batch_size：每个批次的样本数量

- 实现迭代器协议： **iter**(self) 返回迭代器对象本身 **next**(self) 返回下一个批次（列表形式），如果数据遍历完则抛出 StopIteration
- 可选拓展：支持 shuffle 参数，打乱数据顺序后再分批次

三、梯度下降

1.什么是平方和损失函数？它在机器学习中起什么作用？

> 平方和损失函数（又称均方误差，MSE）把模型预测值与真实值之间的差异取平方后求平均，得到一个非负数： 1.公式：MSE = (1/n) Σ(真实值 − 预测值)² 2.作用：在回归任务中衡量模型整体预测精度，数值越小表示误差越小；训练时通过梯度下降等优化算法不断调整模型参数，使 MSE 降到最低，从而找到最优模型。

2.什么是梯度下降？学习率有什么作用？请写出 一元线性函数y=wx+b 对平方和损失函数 L(w)的梯度下降权重更新公式。

> 1. 梯度下降是什么 梯度下降是一种迭代优化算法： 目标：让损失函数 L(参数) 尽可能小。 思路：把参数想象成山顶，损失函数是“高度”。算法像“盲人下山”，每一步都沿着当前最陡的下坡方向（即负梯度方向）走，直到接近谷底（局部最小值）。
> 2. 学习率的作用 学习率 η 就是“每一步迈多大”。η 太大：步子过长，可能直接跨过谷底，甚至发散。 η 太小：步子过短，虽然稳，但需要很多次迭代才能到达谷底。 通常取 0.001～0.1 之间的小正数，并可随迭代逐步减小。
> 3. 一元线性回归的梯度更新公式 模型：y_pred = w·x + b 损失函数（平方和）： L(w,b) = Σ(y_true − y_pred)² 对权重 w 和偏置 b 分别求偏导，得到梯度： ∂L/∂w = −2·Σ(y_true − y_pred)·x ∂L/∂b = −2·Σ(y_true − y_pred) 梯度下降更新规则： w_new = w_old − η·∂L/∂w b_new = b_old − η·∂L/∂b 写成一行就是： w ← w − η·[−2·Σ(y−(wx+b))·x] b ← b − η·[−2·Σ(y−(wx+b))]

3.学习率过大和过小分别会产生什么后果

> 学习率过大 1.震荡或发散：参数更新步长太大，每次迭代都跨过最优解，导致损失函数在高低之间来回跳动，甚至越走越远。 2.不收敛：模型始终无法稳定到最小值，训练过程可能被迫中断仍得不到好结果。 3.梯度爆炸：极端情况下，梯度值迅速增大，出现 NaN 或溢出错误。
> 学习率过小 1.收敛极慢：参数每次只挪一小步，需要成千上万次迭代才能接近最优解，训练时间显著增加。 2.陷入局部最优：步长太小，容易卡在“小坑”里出不来，错过更好的全局最优。 3.资源浪费：在有限计算预算下，可能被迫提前终止，最终得到次优模型。

4.解释梯度下降和随机梯度下降的区别，以及在训练中各自的优缺点。

> 区别： 1.梯度下降每次更新用的数据是整个训练集；随机梯度下降每次更新用的数据是只用1个（或极少量）样本。 2.梯度下降的准确性高，方向稳定；随机梯度下降的准确性低，带有随机噪音。 3.梯度下降的计算开销大，每步都要遍历全部数据，内存和时间成本高；随机梯度下降每步计算量极小，可以在线/实时更新。
> 训练中的优缺点： 梯度下降： 优点：收敛路径平滑，容易调到全局最优；梯度精确，步长可以设得较大。 缺点：数据量大时速度极慢；无法在线学习，新数据加入就得重算全集。 随机梯度下降： 优点：训练快，适合海量数据；随机性有助于跳出局部最优；天然支持在线学习。 缺点：收敛波动性较大，可能在最优值附近震荡；需要更精细的学习率调度，否则易发散。

5.什么是 mini-batch 梯度下降？它与全量梯度下降和 SGD 有什么联系？

> 什么是mini-batch： mini-batch 梯度下降把训练集切成若干“小批量”（典型大小 32、64、128 等），每次迭代只用一个 mini-batch 计算梯度并更新参数。遍历完全部 mini-batch 算作一个 epoch。
> 与全量梯度下降和SGD的联系： 1.全量梯度下降可以看作mini-batch的“极端特例”——batch size等于全部样本。 2.SGD可以视作mini-batch的“另一个极端”——batch size=1.

6.编程实例：手搓梯度下降函数

我们想用线性函数y=wx+b来拟合上述数据，损失函数使用平方和损失函数，请实现函数 要求：

- 每次迭代计算 w 和 b 的梯度并更新参数
- 每次迭代打印当前 w、b 和对应的损失 L(w, b)
- 初始化 w=0, b=0, 学习率 lr=0.1, 迭代 5 次
- 输出每次迭代的 w、b 和损失
- 支持 随机梯度下降，每次随机选择一个样本计算梯度
- 支持 mini-batch，每次选择若干样本计算梯度

四、基于Pytorch的线性回归实现

1.请自行上网查阅相关资料，安装Anaconda或 Miniconda并自行配置如Pytorch、pandas、nump等常见的机器学习相关的库。并请分享你在这个过程中遇到的困难，收获与心得以及取得的成果展示。

2.什么是线性回归？它解决什么问题？一元线性回归和多元线性回归有什么区别？除了回归问题之外，机器学习中还有哪些经典问题类型？

> 线性回归 线性回归是一种用直线（或超平面）描述自变量与因变量之间线性关系的统计方法，核心是找到一条“最佳拟合直线”，使得预测值与真实值之间的均方误差最小。它解决的问题是回归任务：给定一个或多个输入特征，预测一个连续的实数值输出，例如房价、温度、销售额等。
> 一元线性回归与多元线性回归的区别: 一元线性回归只考虑一个自变量，模型形式是一条直线，用来描述单一因素对结果的影响；多元线性回归则同时引入两个或以上的自变量，模型变成一个超平面，能够综合多个因素来预测结果。 由于涉及多个变量，多元回归需要额外关注变量之间的共线性问题，并采用调整后的指标来评估模型优度，而一元回归只需关注单一变量的显著性即可。
> 机器学习中的经典问题类型 机器学习中的其他经典问题类型 分类：预测离散类别标签，如垃圾邮件检测（是/否）、图像识别（猫/狗/汽车）。 聚类：无监督地将数据分组，如客户细分、异常检测。 降维：减少特征数量同时保留主要信息，如 PCA、t-SNE，用于可视化或去噪。 强化学习：智能体通过与环境交互获得奖励信号，学习最优策略，如游戏 AI、机器人控制。

3.编程实例：

请利用下面给出的数据，在尽可能不调用过多高级Pytorch库函数（如torch.nn.Linear、torch.optim.SGD）的前提下，实现下述任务：

- 小批量数据迭代器： 1.编写函数 data_iter(batch_size, features, labels) 2.随机打乱数据，并每次返回一个批次 (x, y)
- 线性回归模型和损失函数： 1.编写函数 linreg(x, w, b) 实现线性预测 2.编写函数 squared_loss(y_hat, y) 计算平方和损失

- 优化器（SGD）： 1.编写函数 sgd(params, lr, batch_size) 2.每步更新参数，并清零梯度
- 训练循环： 1.初始化参数 w 和 b 2.设定超参数：学习率、迭代次数、批量大小 3.使用 for 循环实现训练过程 4.每个 epoch 后打印训练损失
- 可选拓展： 1.训练可视化：使用 Matplotlib 或其他可视化工具绘制每个 epoch 的训练损失变化曲线。 2.超参数调节：在完成基础训练循环后，自行尝试修改参数学习率（lr）、迭代次数（num_epochs）、批量大小（batch_size）并记录不同参数组合下训练损失的变化，总结不同超参数对训练速度和模型收敛效果的影响。