**《数据统计与分析基础实验》**

实验一报告

|  |  |
| --- | --- |
| 班级： | 计224 |
| 姓名： | 王小康 |
| 学号： | 225432 |

**实验一 基本语法及运算**

**一、实验目的**

掌握实用软件（如 Python、C、C++等任选一种）的基本语法、基本数据类型的使用方法，掌握从键盘输入并运算基本类型的数据，熟练运用分支、 循环等语句控制程序流程。

**二、实验内容**

1. 编写程序，记录十名学生的信息，至少包括姓名、年龄、出生年、数据统计分析课程实验成绩，程序输出前n个学生的年龄平均值、数据统计分析课程实验成绩平均值，n为函数的输入参数。
2. 找出1至999之间是13的倍数或者前两位数字是13的数字，输出这些数字，并统计有多少个。
3. 编写成绩转化为绩点的函数，用98，93，89，73，66分别调用函数，生成对应绩点。
4. 随机生成两个长为100且服从标准正态分布的向量，然后将两向量所有偶数位的数值对调，输出所有的4个向量（对调前的2个，对调后的2个）。
5. \*已知XYZ+YZZ=532，其中X、Y和Z为数字，编程求出X，Y和Z的值。

**三、实验源程序及运行结果（可附截图）**

1.程序如下：

class Student:

    def \_\_init\_\_(self, name, age, birth\_year, stats\_grade):

        self.name = name

        self.age = age

        self.birth\_year = birth\_year

        self.stats\_grade = stats\_grade

class StudentRecords:

    def \_\_init\_\_(self):

        self.students = []

    def add\_student(self, name, age, birth\_year, stats\_grade):

        self.students.append(Student(name, age, birth\_year, stats\_grade))

    def calculate\_averages(self, n):

        if n > len(self.students) or n <= 0:

            raise ValueError("n must be between 1 and the number of students.")

        selected\_students = self.students[:n]

        avg\_age = sum(student.age for student in selected\_students) / n

        avg\_stats\_grade = sum(student.stats\_grade for student in selected\_students) / n

        return avg\_age, avg\_stats\_grade

# 示例程序

if \_\_name\_\_ == "\_\_main\_\_":

    # 创建学生记录

    records = StudentRecords()

    # 添加10名学生

    records.add\_student("Alice", 20, 2003, 85)

    records.add\_student("Bob", 21, 2002, 90)

    records.add\_student("Charlie", 22, 2001, 88)

    records.add\_student("Daisy", 20, 2003, 92)

    records.add\_student("Edward", 23, 2000, 75)

    records.add\_student("Fiona", 19, 2004, 95)

    records.add\_student("George", 21, 2002, 87)

    records.add\_student("Helen", 20, 2003, 80)

    records.add\_student("Ian", 22, 2001, 78)

    records.add\_student("Jack", 19, 2004, 85)

    # 输入前n个学生

    try:

        n = int(input("请输入要统计的学生人数 (n): "))

        avg\_age, avg\_stats\_grade = records.calculate\_averages(n)

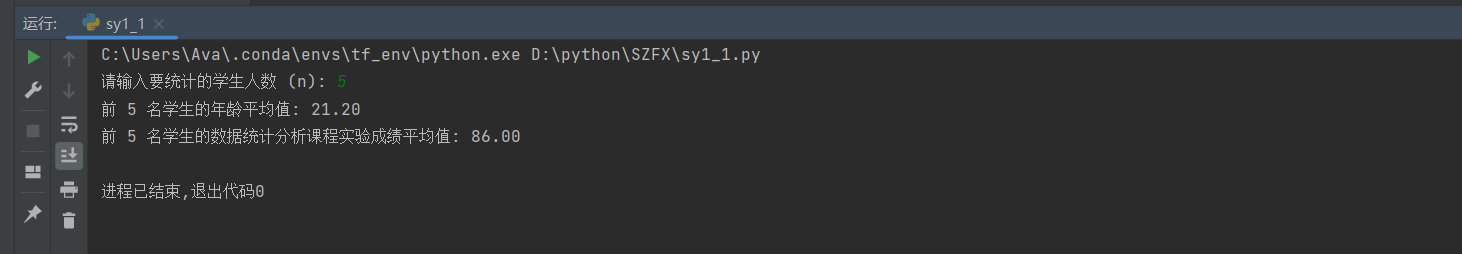
        print(f"前 {n} 名学生的年龄平均值: {avg\_age:.2f}")

        print(f"前 {n} 名学生的数据统计分析课程实验成绩平均值: {avg\_stats\_grade:.2f}")

    except ValueError as e:

        print(f"输入错误: {e}")

结果如下：



2.程序如下：

def find\_numbers():

    results = []

    for num in range(1, 1000):

        # 检查是否是13的倍数

        if num % 13 == 0:

            results.append(num)

        # 检查是否前两位数字是13

        elif str(num).zfill(3)[:2] == "13":

            results.append(num)

    # 输出结果

    print("符合条件的数字如下：")

    print(results)

    print(f"总共有 {len(results)} 个符合条件的数字。")

if \_\_name\_\_ == "\_\_main\_\_":

    find\_numbers()

结果如下：

C:\Users\Ava\.conda\envs\tf\_env\python.exe D:\python\SZFX\sy1\_2.py

符合条件的数字如下：

[13, 26, 39, 52, 65, 78, 91, 104, 117, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 143, 156, 169, 182, 195, 208, 221, 234, 247, 260, 273, 286, 299, 312, 325, 338, 351, 364, 377, 390, 403, 416, 429, 442, 455, 468, 481, 494, 507, 520, 533, 546, 559, 572, 585, 598, 611, 624, 637, 650, 663, 676, 689, 702, 715, 728, 741, 754, 767, 780, 793, 806, 819, 832, 845, 858, 871, 884, 897, 910, 923, 936, 949, 962, 975, 988]

总共有 85 个符合条件的数字。

进程已结束,退出代码0

3.程序如下：

def grade\_to\_gpa(score):

    """

    将成绩转化为绩点的函数。

    :param score: 分数 (0-100)

    :return: 对应的绩点

    """

    if score >= 90:

        return 4.0

    elif score >= 85:

        return 3.7

    elif score >= 80:

        return 3.3

    elif score >= 75:

        return 3.0

    elif score >= 70:

        return 2.7

    elif score >= 65:

        return 2.3

    elif score >= 60:

        return 2.0

    else:

        return 0.0

# 调用函数并生成绩点

scores = [98, 93, 89, 73, 66]

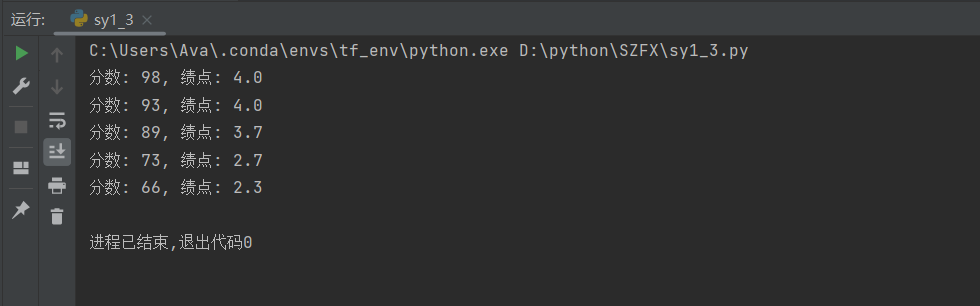
gpas = [grade\_to\_gpa(score) for score in scores]

# 输出结果

for score, gpa in zip(scores, gpas):

    print(f"分数: {score}, 绩点: {gpa:.1f}")

结果如下：



4.程序如下：

import numpy as np

# 生成两个服从标准正态分布的向量

vector1 = np.random.normal(loc=0, scale=1, size=100)

vector2 = np.random.normal(loc=0, scale=1, size=100)

# 复制向量用于对调后的结果

vector1\_swapped = vector1.copy()

vector2\_swapped = vector2.copy()

# 对偶数位进行对调 (Python索引从0开始，偶数位为索引1, 3, 5,...)

vector1\_swapped[1::2], vector2\_swapped[1::2] = vector2[1::2], vector1[1::2]

# 输出结果

print("原始向量1：")

print(vector1)

print("\n原始向量2：")

print(vector2)

print("\n对调后向量1：")

print(vector1\_swapped)

print("\n对调后向量2：")

print(vector2\_swapped)

结果如下：

C:\Users\Ava\.conda\envs\tf\_env\python.exe D:\python\SZFX\sy1\_4.py

原始向量1：

[ 0.73764902 -0.86924308 0.17527582 -0.9914442 1.64905815 -1.30316124

0.92590793 -0.7348526 0.76823581 0.63638274 -1.16533773 -0.5847355

1.04333493 0.27341888 -0.7831909 -0.44547645 1.43975118 1.19978955

1.30610537 0.04536494 -1.00492294 0.79582114 1.82417735 0.58236859

1.11014521 0.00436767 0.28930124 0.18840303 0.2339275 0.36118782

1.52698178 -0.13141611 2.82093324 1.90271487 0.32554739 0.04248717

0.95235415 -1.62838361 0.95435448 -0.80759988 1.01501697 -0.31152627

0.9120414 -1.37130754 0.22358833 -0.73922384 -2.60404718 -0.53757941

-0.11565289 3.02262216 -0.4150234 -1.23067405 0.46275734 -0.77648623

0.05723359 1.20792461 0.54244708 0.50019012 -2.66032326 -0.4784051

-0.4118497 -1.54574312 0.78124182 -0.17370366 -2.16993433 0.66125474

2.70672543 -1.29228982 0.157666 0.65485342 0.085926 -0.40162111

0.449146 0.22280979 -3.18266753 0.97745579 0.82297442 0.53776238

1.6848502 0.61506772 1.92377477 0.26418542 0.08567521 -0.37406047

0.56923995 -0.74685377 -0.87254789 0.85480737 0.2769859 0.43018241

0.57138967 -2.19984932 -0.07142912 0.05768144 -0.94572951 1.02215847

0.09469875 0.42704646 0.10237625 -0.39870841]

原始向量2：

[-0.43711109 -0.11784754 0.8054026 -1.28010894 -0.27132674 -0.05651238

0.30107912 1.04382734 -0.04360707 -0.15330979 0.3992112 0.07008631

-0.16973846 -0.36508803 -0.63430609 0.1726185 0.79755814 -1.50110139

0.1769625 0.23857449 0.31582664 0.79319287 1.03108994 -1.7017987

0.59218901 0.24099913 1.10849956 0.72511533 -1.79336948 0.89519346

0.04321196 1.02611742 -0.61924047 0.53295281 0.52508204 -1.43642701

1.17205927 -0.86634097 0.45273026 -1.15853529 1.33585747 0.02313867

0.30802657 -0.28840601 -0.70845609 1.23954156 -0.40173899 1.63765519

0.39075656 1.30877109 -0.81897155 -0.50658403 1.93145277 1.84751129

-0.55077996 1.53982615 1.31555487 -0.36035384 0.47043969 -1.19763105

0.24878371 1.26579959 -0.29266675 2.09551012 0.80716003 0.28316783

0.61562319 -2.60715774 -0.08243267 -0.37063448 0.21754989 0.45195974

-0.78238638 1.07344504 -1.37744099 -0.59191205 -1.97283199 -0.6409099

1.34459544 -1.05477772 -0.75758852 -1.24228785 0.79503164 -0.00511297

0.55484984 0.93392868 0.16547866 0.98833104 0.05155941 -1.06521335

-1.37791394 0.0896006 -1.46316463 -1.01377059 -1.90950695 0.18111639

0.95825248 1.33551191 2.58444416 -0.62305916]

对调后向量1：

[ 0.73764902 -0.11784754 0.17527582 -1.28010894 1.64905815 -0.05651238

0.92590793 1.04382734 0.76823581 -0.15330979 -1.16533773 0.07008631

1.04333493 -0.36508803 -0.7831909 0.1726185 1.43975118 -1.50110139

1.30610537 0.23857449 -1.00492294 0.79319287 1.82417735 -1.7017987

1.11014521 0.24099913 0.28930124 0.72511533 0.2339275 0.89519346

1.52698178 1.02611742 2.82093324 0.53295281 0.32554739 -1.43642701

0.95235415 -0.86634097 0.95435448 -1.15853529 1.01501697 0.02313867

0.9120414 -0.28840601 0.22358833 1.23954156 -2.60404718 1.63765519

-0.11565289 1.30877109 -0.4150234 -0.50658403 0.46275734 1.84751129

0.05723359 1.53982615 0.54244708 -0.36035384 -2.66032326 -1.19763105

-0.4118497 1.26579959 0.78124182 2.09551012 -2.16993433 0.28316783

2.70672543 -2.60715774 0.157666 -0.37063448 0.085926 0.45195974

0.449146 1.07344504 -3.18266753 -0.59191205 0.82297442 -0.6409099

1.6848502 -1.05477772 1.92377477 -1.24228785 0.08567521 -0.00511297

0.56923995 0.93392868 -0.87254789 0.98833104 0.2769859 -1.06521335

0.57138967 0.0896006 -0.07142912 -1.01377059 -0.94572951 0.18111639

0.09469875 1.33551191 0.10237625 -0.62305916]

对调后向量2：

[-0.43711109 -0.86924308 0.8054026 -0.9914442 -0.27132674 -1.30316124

0.30107912 -0.7348526 -0.04360707 0.63638274 0.3992112 -0.5847355

-0.16973846 0.27341888 -0.63430609 -0.44547645 0.79755814 1.19978955

0.1769625 0.04536494 0.31582664 0.79582114 1.03108994 0.58236859

0.59218901 0.00436767 1.10849956 0.18840303 -1.79336948 0.36118782

0.04321196 -0.13141611 -0.61924047 1.90271487 0.52508204 0.04248717

1.17205927 -1.62838361 0.45273026 -0.80759988 1.33585747 -0.31152627

0.30802657 -1.37130754 -0.70845609 -0.73922384 -0.40173899 -0.53757941

0.39075656 3.02262216 -0.81897155 -1.23067405 1.93145277 -0.77648623

-0.55077996 1.20792461 1.31555487 0.50019012 0.47043969 -0.4784051

0.24878371 -1.54574312 -0.29266675 -0.17370366 0.80716003 0.66125474

0.61562319 -1.29228982 -0.08243267 0.65485342 0.21754989 -0.40162111

-0.78238638 0.22280979 -1.37744099 0.97745579 -1.97283199 0.53776238

1.34459544 0.61506772 -0.75758852 0.26418542 0.79503164 -0.37406047

0.55484984 -0.74685377 0.16547866 0.85480737 0.05155941 0.43018241

-1.37791394 -2.19984932 -1.46316463 0.05768144 -1.90950695 1.02215847

0.95825248 0.42704646 2.58444416 -0.39870841]

进程已结束,退出代码0

5.程序如下：

# 遍历X, Y, Z的所有可能取值

for X in range(1, 10):  # X不能为0，因为它是一个数字的最高位

    for Y in range(0, 10):

        for Z in range(0, 10):

            # 计算XYZ和YZZ的值

            XYZ = 100 \* X + 10 \* Y + Z

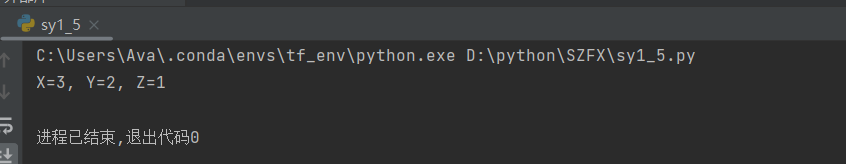
            YZZ = 100 \* Y + 10 \* Z + Z

            # 检查是否满足XYZ + YZZ = 532

            if XYZ + YZZ == 532:

                print(f"X={X}, Y={Y}, Z={Z}")

结果如下：



**四、实验感想**

通过这次实验，我了解了如何将数学问题用代码表达。。此外，本次实验也让我体会到代码逻辑清晰的重要性，能够使后续的调试和扩展更为便捷。本次实验不仅让我加深了对 Python 基础编程的理解，还让我认识到数学与编程的紧密结合如何帮助解决实际问题。在未来的学习中，我将尝试更复杂的问题解决，并学习更优化的算法设计方法。也使我掌握实用软件的基本语法、基本数据类型的使用方法，掌握了从键盘输入并运算基本类型的数据，熟练运用分支、 循环等语句控制程序流程。