第二次上机报告

> 2233316027 王博想

1. 输出九九乘法表

问题重述

按如下形式打印九九乘法表

$$
\begin{aligned}
&1\times 1=1\\
&1\times 2=2\ \ 2\times 2=4\\
&1\times 3=3\ \ 2\times 3=6\ \ 3\times 3=9\\
&1\times 4=4\ \ 2\times 4=8\ \ 3\times 4=12\ \ 4\times 4=16\\
&1\times 5=5\ \ 2\times 5=10\ \ 3\times 5=15\ \ 4\times 5=20\ \ 5\times 5=25\\
&1\times 6=6\ \ 2\times 6=12\ \ 3\times 6=18\ \ 4\times 6=24\ \ 5\times 6=30\ \ 6\times 6=36\\
&1\times 7=7\ \ 2\times 7=14\ \ 3\times 7=21\ \ 4\times 7=28\ \ 5\times 7=35\ \ 6\times 7=42\ \ 7\times 7=49\\
&1\times 8=8\ \ 2\times 8=16\ \ 3\times 8=24\ \ 4\times 8=32\ \ 5\times 8=40\ \ 6\times 8=48\ \ 7\times 8=56\ \ 8\times 8=64\\
&1\times 9=9\ \ 2\times 9=18\ \ 3\times 9=27\ \ 4\times 9=36\ \ 5\times 9=45\ \ 6\times 9=54\ \ 7\times 9=63\ \ 8\times 9=72
\end{aligned}
$$

实现思想

利用列表推导式， `for i in range(1,10)` 控制行数，从 1 到 9； `for j in range(1,i+1)` 控制每一行的列数，确保 `j≤i`，实现上三角形排列。

接着使用 `f-string` 格式化字符串， `f"\n{j}x{i}={i*j} "` 生成形如 `\n2x3=6 ` 的字符串。

对于换行 `\n` 要保证只在 `j=1` 且 `i≠1` 时生效，即每行首个表达式前换行。于是使用切片操作 `[j>1:]`： `j>1` 为 `True` 时，转换为 `1`，跳过 `\n`，确保只有第一列换行； `[i<2:]`： `i<2` 仅在 `i=1` 为 `True`，即 `i=1` 时不去掉 `\n`，保持第一行正确。

最后利用 `''.join()` 将所有字符串连接并打印，实现了格式整齐的九九乘法表输出。

源代码

```python
print(''.join(f"\n{j}x{i}={i*j} "[j>1:][i<2:] for i in range(1,10)
for j in range(1,i+1)))
```

结果及说明

```
1x1=1
1x2=2 2x2=4
1x3=3 2x3=6 3x3=9
1x4=4 2x4=8 3x4=12 4x4=16
1x5=5 2x5=10 3x5=15 4x5=20 5x5=25
1x6=6 2x6=12 3x6=18 4x6=24 5x6=30 6x6=36
1x7=7 2x7=14 3x7=21 4x7=28 5x7=35 6x7=42 7x7=49
1x8=8 2x8=16 3x8=24 4x8=32 5x8=40 6x8=48 7x8=56 8x8=64
1x9=9 2x9=18 3x9=27 4x9=36 5x9=45 6x9=54 7x9=63 8x9=72 9x9=81
```

2. 求函数极限

问题重述

用程序计算方法求下列函数的极限

$$\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^x \qquad \lim_{x\to 0}\sin\frac{1}{x}$$

实现思想

代码首先使用 `sympy` 作为符号计算库，定义变量 `x` 并构造表达式。对于第一个极限问题 $\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^x$，代码利用 `sympy.limit` 计算极限，返回值为 `e`，符合数学推导结果。第二个极限问题 $\lim_{x\to 0}\sin\frac{1}{x}$ 由于 $\sin(1/x)$ 在 0 处无极限，`sympy` 不会返回固定值，表明极限不存在。最终，代码打印计算结果，验证极限的数学性质。

源代码

```python
import sympy as sp

def limit_exp():
    x = sp.symbols('x')
    expr = (1 + 1/x)**x
    lim = sp.limit(expr, x, sp.oo)
    return lim

def limit_sin():
```

```python
    x = sp.symbols('x')
    expr = sp.sin(1/x)
    lim = sp.limit(expr, x, 0)
    return lim

exp_limit = limit_exp()
sin_limit = limit_sin()

print("lim (1 + 1/x)^x as x → inf:", exp_limit)
print("lim sin(1/x) as x → 0:", sin_limit)
```

结果及说明

```
lim (1 + 1/x)^x as x → inf: E
lim sin(1/x) as x → 0: AccumBounds(-1, 1)
```

结果表明第一个极限的值是 e，而第二个极限不存在，在 $[-1, 1]$ 之间震荡

3. Gauss 消元法

问题重述

编写高斯消元法求解以下方程组

$$(1)\begin{cases} x_1 + 0.5x_2 + 0.33x_3 = 1.83 \\ 0.5x_1 + 0.33x_2 + 0.25x_3 = 1.08 \\ 0.33x_1 + 0.25x_2 + 0.2x_3 = 0.783 \end{cases}$$

$$(2)\begin{cases} x_1 + 0.5x_2 + 0.33x_3 + 0.25x_4 = 2.08 \\ 0.5x_1 + 0.33x_2 + 0.25x_3 + 0.2x_4 = 1.28 \\ 0.33x_1 + 0.25x_2 + 0.2x_3 + 0.17x_4 = 0.95 \\ 0.25x_1 + 0.2x_2 + 0.167x_3 + 0.143x_4 = 0.76 \end{cases}$$

实现思想

该代码使用高斯消元法求解两个线性方程组。

首先，代码定义了 `gaussian_elimination` 函数，该函数接收系数矩阵 `A` 和常数向量 `b` 作为输入。函数首先进行主元选取，即在当前列找到最大元素并交换所在行，以减少舍入误差。随后进行消元操作，将当前列以下的元素变为零，使矩阵变为上三角形。消元完成后，通过回代求解 x，从最后一行开始逐步向上计算。

对于两个方程组，代码分别定义 A1 和 A2 作为系数矩阵，并调用 `gaussian_elimination` 计算解并打印。

源代码

```python
import numpy as np

def gaussian_elimination(A, b):
    n = len(b)
    for i in range(n):
        max_row = i + np.argmax(np.abs(A[i:n, i]))
        if max_row != i:
            A[[i, max_row]] = A[[max_row, i]]
            b[i], b[max_row] = b[max_row], b[i]

        for j in range(i+1, n):
            factor = A[j, i] / A[i, i]
            A[j, i:] -= factor * A[i, i:]
            b[j] -= factor * b[i]

    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (b[i] - np.dot(A[i, i+1:], x[i+1:])) / A[i, i]

    return x

A1 = np.array([[1, 0.5, 0.33], [0.5, 0.33, 0.25], [0.33, 0.25, 0.2]],
dtype=float)
b1 = np.array([1.83, 1.08, 0.783], dtype=float)
solution1 = gaussian_elimination(A1, b1)
print("Solution for system (1):", solution1)

A2 = np.array([[1, 0.5, 0.33, 0.25], [0.5, 0.33, 0.25, 0.2], [0.33,
0.25, 0.2, 0.17], [0.25, 0.2, 0.167, 0.143]], dtype=float)
b2 = np.array([2.08, 1.28, 0.95, 0.76], dtype=float)
solution2 = gaussian_elimination(A2, b2)
print("Solution for system (2):", solution2)
```

结果及说明

```
Solution for system (1): [ 1.76666667 -3.04761905 4.80952381]
Solution for system (2): [1. 1. 1. 1.]
```

结果表明第一个方程组的解是：$x_1 \approx 1.7667,\ x_2 \approx -3.0476,\ x_3 \approx 4.8095$

第二个方程组的解是：$x_1 = x_2 = x_3 = x_4 = 1$