面向对象第一单元第一次作业指导书

摘要

本次作业，需要完成的任务为**简单多项式导函数**的求解。

问题

设定

首先是一些基本概念的声明：

- **带符号整数**：支持前导0的**十**进制带符号整数，符号可省略，无进制标识，如：`+02`、`-16`、`19260817`等。
- **幂函数**
  - **一般形式** 由自变量x和指数组成，指数为一个带符号整数，如：`x ** +2`。
  - **省略形式** 当指数为1的时候，可以采用省略形式，如：`x`。
- **项**
  - **变量项**
    - 系数与幂函数的乘积，且系数位于幂函数前，系数为一个带符号整数，如：`2 * x ** 2`、`-1 * x`。
    - 系数为1的时候，可以省略系数或表示为正号开头的形式，如：`x ** 2`、`+ x ** 2`。
    - 系数为-1的时候，可以表示为负号开头的形式，如：`-x ** 2`。
  - **常数项** 包含一个带符号整数，如：`233`。
- **表达式** 由加法和减法运算符连接若干项组成，如：`-1 + x ** 233 - x ** 06`。此外，**在第一项之前，可以带一个正号或者负号**，如：`- -1 + x ** 233`、`+ -2 + x ** 19260817`。注意，**空串不属于合法的表达式。**
- **空白字符** 在本次作业中，空白字符包含且仅包含空格`<space>`（ascii值32）和水平制表符`\t`（ascii值9）。其他的空白字符，均属于非法字符。

  对于空白字符，有以下几点规定：
  - 带符号整数内不允许包含空白字符，注意**符号与整数之间**也不允许包含空白字符。
  - 指数运算符内不允许包含空白字符，如`* *`不合法。
  - 幂函数、项、表达式，在不与前两条条矛盾的前提下，可以在任意位置包含任意数量的空白字符。

设定的形式化表述

- 表达式 → 空白项 [加减 空白项] 项 空白项 | 表达式 加减 空白项 项 空白项
- 项 → 变量项|常数项
- 变量项 → [带符号的整数 空白项 * 空白项] 幂函数
- 常数项 → 带符号的整数
- 幂函数 → x [空白项 指数]
- 指数 → ** 空白项 带符号的整数
- 带符号的整数 → [加减] 允许前导零的整数
- 允许前导零的整数 → {0|1|2|...|7|8|9}
- 空白项 → {空白字符}

- 空白字符 → ` `（空格） | `\t`
- 加减 → + | -

其中{}表示0个、1个或多个，[]表示0个或1个，|表示多个之中选择。

式子的具体含义参照其数学含义。

若输入字符串能够由 “表达式” 推导得出，则为正确输入字符串。具体推导方法请参阅 “第一单元形式化表述说明” 文档。

描述

求导是数学计算中的一个计算方法，它的定义就是，当自变量的增量趋于零时，因变量的增量与自变量的增量之商的极限。

在本次作业中，我们要对输入的多项式进行求导计算，化简，并输出它的导函数。

本次作业可能用到的求导公式是

$$\text{Ⅰ.当}\ f(x)=a\ \text{（}a\text{为常数）时，}\ f'(x)=0$$

$$\text{Ⅱ.当}\ f(x)=ax^{b}\ \text{时，}\ f'(x)=abx^{b-1}$$

例如：

$$\text{当}\ f(x)=2x^{6}+6x^{4}\ \text{时，}\ f'(x)=12x^{5}+24x^{3}$$

输入为`2*x**6+6*x**4`，输出为`12*x**5+24*x**3`。

一些规定

- 一个表达式可能有多个解释。因此，对于一个表达式，只要存在一条合法解释，该表达式即为合法，且我们保证以这些解释在数学意义上均相等。

评测

输入模式

输入中，包含且仅包含一行，表示一个表达式。

本次作业保证输入数据全部为合法（格式正确）的表达式，不需要进行格式检查！！！

输出模式

关于输出，由于本次作业输入数据全部为合法的表达式，因此程序不需要对输入数据的合法性进行判定

应当输出一行，表示求出的导函数。格式同样需要满足上述的表达式基本格式规则，其中要求带符号的整数均为**十进制形式**。

数据限制

- **数据的最大长度为1000**（请注意，这里不是有效长度，是去除右侧换行符后的总长度），并且是**合法的**表达式。

上述限制被定义为**数据基本限制**。在测试中，不会出现不符合该限制的数据。在此范围限制内，不作其他任何限制。

判定模式

正确性判定

对于这次作业结果正确性的判定，在输出符合格式要求的前提下，我们采用如下的方式：

- **在区间**$[-10,10]$**上，进行1000次线性随机选取**，设得到的数为$\{x_i\}$ $(1 \leq i \leq 1000)$
- 设输入多项式为$f(x)$，其导函数为$f'(x)$（即正确答案，由sympy进行符号运算），将$\{x_i\}$依次代入$f'(x)$，得到结果$\{a_i\}$
- 设待测输出的多项式为$g'(x)$，将$\{x_i\}$依次代入$g'(x)$，得到结果$\{b_i\}$
- 将数列$\{a_i\}$和数列$\{b_i\}$依次进行比较，判定每个数是否依次相等
- **如果全部相等，则认为该组输出正确**，否则认为错误

其中

- 在比较两个数的时候，**判定是否相等的依据**是：对于数a_i表示正确结果，b_i表示根据输出计算出的结果，若满足

$$\frac{|a_i - b_i|}{\max(|a_i|, 1)} < 10^{-8}$$

则视为a_i与b_i相等。

- 考虑到可能会出现随机出的数位于无意义点上导致计算出错，故在上述计算b_i的过程中，**如果第**i**个数计算错误，则将重新生成该**x_i**，并重新计算。**每个数最多将会重试五次，如果重试次数达到上限后依然无法正常计算，则判定该组输出错误。**举例说明的话，就是表达式**$\frac{x^2}{x}$**最终也会被判定为和表达式**x**等价。**(由a_i不可计算引发的重新计算将不被计算在这五次内)

简而言之，可以理解为：**只要是和标准结果等价的表达式（允许定义域上的点差异），都会被认定为正确答案。**

性能分判定

在本次作业中，性能分的唯一评判依据，是输出结果的有效长度。

有效长度定义为，输出结果去除所有的空白字符（`<space>`、`\t`）后的长度，设为L。

设某同学给出的正确答案的有效长度为L_p，所有人目前给出的正确答案的有效长度的最小值为L_{min}。

设$x = \frac{L_p}{L_{min}}$，则该同学性能分百分比为：

$$r(x) = 100\% \cdot \begin{cases} 1 & x = 1 \\ -31.8239x^4 + 155.9038x^3 - 279.2180x^2 + 214.0743x - 57.9370 & 1 < x \leq 1.5 \\ 0 & x > 1.5 \end{cases}$$

简单来说，就是这样：

x	$r(x)$
1.0	100.0%
1.05	79.9%
1.1	60.5%
1.2	29.0%
1.3	10.9%
1.4	4.5%
1.5	0.0%

值得注意的是，**获得性能分的前提是，在正确性判定环节被判定为正确。**如果被判定为错误，则性能分部分为0分。

互测相关

在互测中，数据必须符合**数据基本限制**，因此：

- 被测程序应当给出正确的求导答案。
- 如果不满足上述数据基本限制的话，则该数据将被系统忽略，不会对被测程序进行测试。

样例

#	输入	输出	解释
1	1	0	根据表达式定义可得。
2	4*x+x**2+x	2*x+5	根据表达式定义可得。
3	4*x+x**2+x	4+2*x+1	未合并同类项，但表达式依然等价。
4	- -4*x + x ** 2 + x	2*x+5	-4x为合法项，且表达式第一项前也可以包含正负号。
5	+4*x - -x**2 + x	2*x+5	-x**2为合法项。
6	+19260817*x	19260817	根据表达式定义可得。
7	+ 19260817*x	19260817	多项式第一项前可以带有正负号。
8	+ +19260817*x	19260817	+19260817*x为合法项，开头可以带有正负号。

注意：由于本作业可被判定为正确的答案不唯一，所以以上样例的输出**仅保证正确性，但并不一定为性能最优解。**

补充信息

关于评测

- 评测时，会自动忽略掉行末的空格以及文件末多余的回车。
- 对于输入，如果包含多行，则忽略第一行以后的内容即可。
- 类似的，对于输出结果，如果包含多行，则在评判的时候将忽略第一行以后的内容。（~~也就是说，你们可以在正文之后附加一些其他的信息以改善自己调试的体验~~）

一点点的提示

- Java内的原生整数类型有`long`和`int`，长度分别为64位和32位。
- 如果觉得上述数据类型不够用的话，可以百度一下Java内可以怎样快速处理这个问题。
- 在Java内，不建议使用静态数组。推荐使用`ArrayList`、`HashMap`、`HashSet`一类的数据结构，快速管理和调配手中无序的数据。
- 关于输入字符串的处理，推荐使用**正则表达式**，相关的 API 可以了解 Matcher 和 Pattern 类。

一点点想说的话

- **不要重复造轮子！不要重复造轮子！不要重复造轮子！**~~重要的事情说三遍~~
- 我们鼓励大家通过Baidu、Google、Stackoverflow等方式自行学习和解决问题。

- 如果还有更多的问题，请到讨论区提问。但是**请善用讨论区**，并在此之前认真阅读包括但不限于课程要求文档、指导书、搜索引擎结果等的内容。