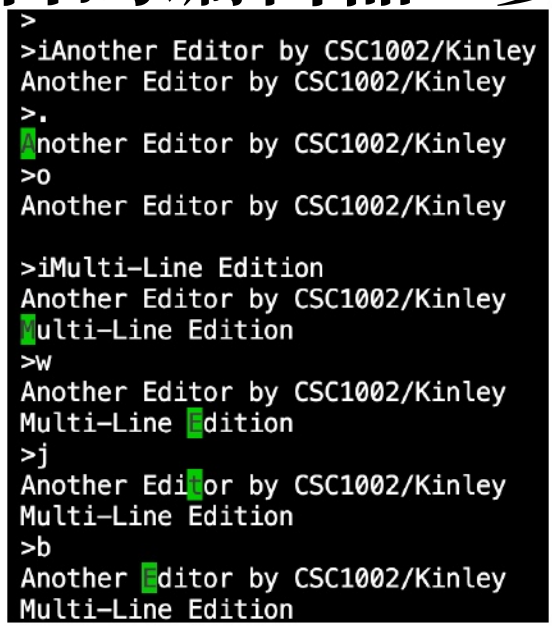
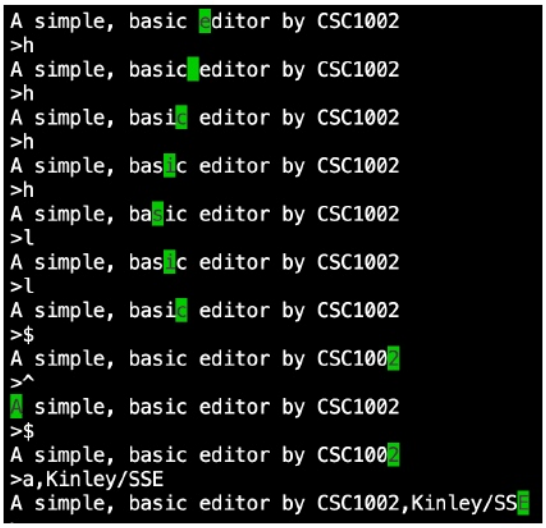
基于控制台的编辑器 - 多行 - 2025



概览

在这次作业中，您将设计并开发一个简单的基于控制台的编辑器，以支持多行编辑。与诸如 Microsoft Word 这类现代高级编辑器不同，后者为用户提供了一个复杂的编辑环境，利用图形屏幕的高分辨率以及鼠标和键盘来定位和调整屏幕上显示的任何文本和图形，从而为我们带来所见即所得（WYSIWYG）的体验。

在早期，由于缺乏丰富的图形显示和鼠标，编辑器的功能有限，通常只提供基于控制台的简单用户界面。编辑操作是通过键盘输入简单的文本命令来完成的，比如插入（i）和追加（a）文本字符串，将编辑器光标向左移动一个字符位置（h），向右移动一个字符位置（l），向前移动一个单词位置（w），向后移动一个单词位置（b）等等。



范围

1. 完成以下所有编辑器命令：

```
? – display this help info
. – toggle row cursor on and off
; – toggle line cursor on and off
h – move cursor left
j – move cursor up
k – move cursor down
l – move cursor right
^ – move cursor to beginning of the line
$ – move cursor to end of the line
w – move cursor to beginning of next word
b – move cursor to beginning of previous word
i – insert <text> before cursor
a – append <text> after cursor
x – delete character at cursor
dw – delete word and trailing spaces at cursor
yy – copy current line to memory
p – paste copied line(s) below line cursor
P – paste copied line(s) above line cursor
dd – delete line
o – insert empty line below
O – insert empty line above
u – undo previous command
r – repeat last command
s – show content
q – quit program
```

   · 注意：有关特定要求的更多信息，请参阅“具体规格”部分；有关程序设计和实现所依据的任何条件，请参阅“假设”部分。

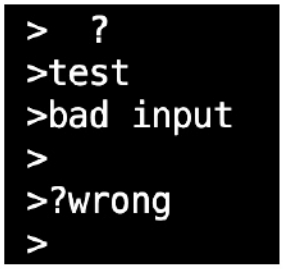
2. 区分大小写的命令 - 所有编辑器命令均区分大小写，例如，大写字母“A”不等于小写字母“a”。

3. 命令类型 - 大多数命令为单字母（小写）命令（例如？ 、$、x、^ 等），而有些是双字母（例如 dw）。大多数命令不需要额外输入，但有少数命令需要，比如插入（i）和追加（a）。

4. 命令提示符（>） - 该提示符为单字符字符串“>”，通过调用 input（'>'） 实现。请参见第一页的截图。

5. 命令语法 - “命令[文本]”，其中“命令”为步骤 1 中所示的命令之一，“文本”仅适用于需要额外输入的命令，例如插入（i）和追加（a）。“< >”括号内的子字符串表示需要额外输入“文本”的命令。“行号”是一个整数值，表示将行光标位置设置到的行号。

6. 命令解析 - 用户输入单个命令，然后按回车键继续。根据“命令语法”解析每个命令字符串，以确保输入字符串与步骤 1 中的某个命令完全匹配，包括所需的额外输入“文本”。当输入无效时，只需显示如下截图所示的提示。



a. 有效的命令输入示例：

“美元符号”

2。 “^”

3。 “h”

iv. “l”

v. “你好，世界”

vi. “你好，世界”

b. 无效命令输入示例：

“美元符号”

2。 “? ”

3。 “? ” “

iv. “啊，你好，世界”

v. “我”

7. 命令执行 - 编辑器将反复提示用户输入编辑器命令，根据“命令语法”解析输入内容，执行命令（如果有效），然后在显示控制台上输出最新的编辑器内容（即使内容没有变化）（除了“? ” 和“q”命令，见下文注释）。在显示编辑器内容之后，编辑器将在新的一行显示另一个提示符（“>”）。更多示例请参阅“示例输出”部分。

注意：当输入帮助命令（？）时，仅输出如步骤 1 所示的帮助菜单；当输入退出命令（‘q’）时，终止程序。

注意：

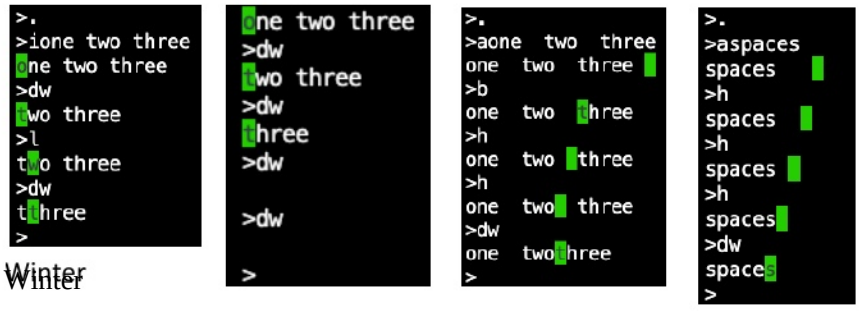
- 请将您的源代码保存在一个单独的文件中。
- 仅使用“许可模块”部分中指定的 Python 模块。
- 在您的设计中仅使用函数，换句话说，不要使用您自己的类对象。

  此外，在另一个函数内部定义的子函数的代码行将被计入父函数的代码行数。

  注意：若未遵循作业说明中所列出的指示，代码风格得分将减少 50%。

~~特定的；具体的；明确的~~

1. 编辑器内容 - 编辑器会显示其内容（如有），这些内容由一个或多个插入/追加/粘贴命令构建而成，以单行或多行文本字符串的形式呈现。如果启用了行光标，则会以绿色等颜色显示其位置。如果启用了列光标，则内容会向右移动一个空格，以“*”符号显示列光标。当编辑器程序首次启动时，其内容为空。更多示例请参阅“示例输出”部分。
2. 行光标 - 它用于在当前行不为空时显示光标所在位置。换句话说，光标将出现在包括空格在内的可打印字符上。光标通过将一个字符用一对转义字符字符串（例如 “\033[42m” 和 “\033[0m”）包裹起来显示。例如，给定字符串 “hello world”，要在字母 ‘e’ 的位置显示绿色光标，要打印的字符串为："h" + "\033[42m" + "e" + "\033[0m" + "llo world"。
3. 插入 - 给定的字符串“Text”将被插入到光标的左侧，光标位置将变为“Text”字符串的开头。
4. 追加 - 给定的字符串“Text”将被插入到当前光标位置的右侧，并且光标位置将移动到“Text”字符串的末尾。
5. 删除单词 - 从光标位置开始删除所有字符，直至下一个单词的开头或行尾。当所有字符都被删除后，该行将为空。请参阅以下示例图。



6. 删除字符 - 删除光标所在处的字符。当所有字符都被删除后，该行将为空。

7. 光标左移和右移 - 当将光标向左或向右移动时，移动一个或多个位置，如果光标已经在最左端或最右端位置，则保持光标不动。

8. 撤销 - 它用于反向执行上一个有效的命令，包括对编辑器内容和/或光标（行、列）位置所做的任何更改。一旦某个命令被撤销，它就不再可用于未来的撤销和重复操作。如果连续执行多个撤销命令，每个命令都会按原始执行顺序的相反顺序撤销一个命令。例如，假设最后两个有效的命令是“ahello”后跟“a world”，那么第一个撤销命令将撤销“a world”，第二个连续的撤销命令将撤销“ahello”。像“s”和“？”这样的命令不能撤销，因为它们不会对编辑器内容进行任何更改，重复命令也是如此。请参阅下图以作说明。

9. 重复 - “重复”命令用于重新执行上一条有效的命令，它为用户省去了重新输入的麻烦。命令“u”、“？”和“s”不能用于“重复”命令，因为这些命令不会直接更新编辑器的内容。例如，考虑以下命令序列：“ahello”，“a world”，“？”，“s”和“u”。如果随后多次输入命令“r”，每次“重复”命令都将始终重新执行“ahello”。有关另一个示例，请参阅“撤销后重复”步骤。

10. 撤销后跟重复 - 在这种情况下，“撤销”不被视为最后一条命令，“重复”命令用于针对“撤销”命令之前的那条命令，而非最近执行的操作。在触发“重复”命令时，会重新执行“撤销”之前输入的任何命令。请参阅下图以作说明。

```
>ahello
hello
>a world
hello world
>u
hello
>r
hellohello
>
```

11. b 命令 - 它将光标移动到光标左侧的单词开头，前提是存在这样的单词；如果不存在这样的单词，则光标保持不动。如果光标位于单词内部，则将其置于该单词的首字母处。请参阅以下图示。

```
>.
>aone  two  three
one  two  three
>h
one  two  three
>b
one  two  three
>h
one  two  three
>b
one  two  three
>b
one  two  three
>
```

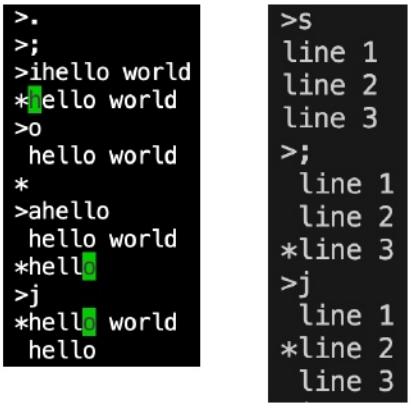
12. w 命令 - 它将光标移动到光标右侧的第一个单词的开头，前提是存在这样的单词；如果不存在这样的单词，则光标保持不动。

13. 单词——单词被定义为连续字符的序列，包括标点符号但不包括空格，换句话说，任何一组没有空格的字符都被视为一个单词，即使其中包含标点符号。请参见下图以作说明。

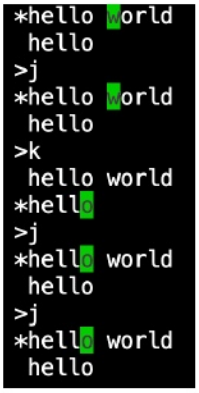
```
>aone-two   three,  "four"
one-two   three,  "four"
>.
one-two   three,  "four"
>b
one-two   three,  "four"
>b
one-two   three,  "four"
>b
one-two   three,  "four"
>dw
three,  "four"
>dw
"four"
>
```

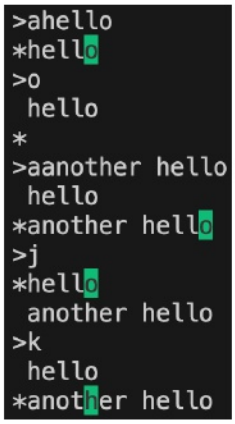
14. 行复制 - 它用于复制或复制当前行的内容，包括新插入的空行。复制的行随后会使用粘贴命令粘贴到编辑器内容中。每次仅保留一条复制的行，换句话说，后续的“行复制”操作会用当前行的内容替换缓存的行。当编辑器内容为空时，“行复制”命令不会执行任何操作。

15. 行光标 - 它用于通过在当前编辑行前加上字符“*”来显示该行，而其他行则仅显示一个空格。请参阅以下插图。

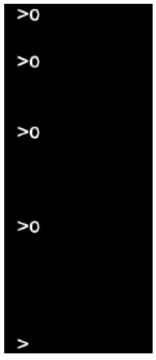


16. 光标上下移动 - 当将光标移至上一行或下一行时，如果该行长度较短，则将光标移至该行末尾（类似于“$”）。如果光标已在第一行或最后一行，则不要使光标从顶部跳至底部或从底部跳至顶部，换句话说，保持行光标所在位置不变。请参考以下图示。

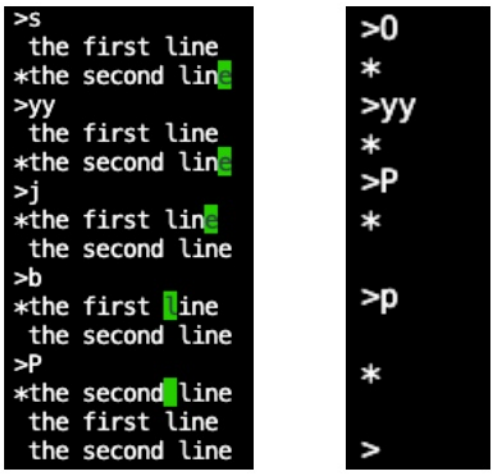




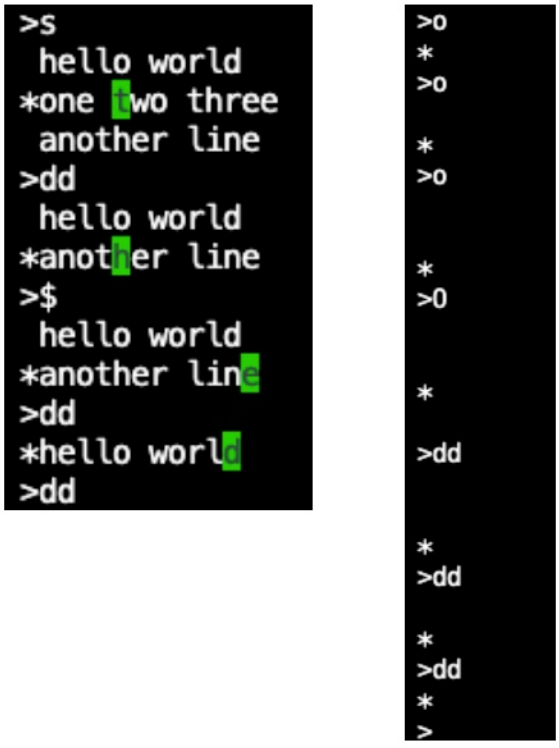
17. 在上一行/下一行插入空行 - 它会在当前行的上方或下方插入一个空行，并相应地调整行光标的位置。请参阅以下插图。



18. “在上一行/下一行粘贴” - 它会将上次复制的行粘贴到当前行的上方或下方，并相应地调整行光标。调整行光标的位置，就好像它是从当前行向上或向下移动到粘贴的行一样。请参考以下图示。



19. 行删除 - 删除当前行，并相应调整行和列光标的位置。请参阅以下插图以作说明。



假设———

1. 本次作业的目标在于展示“问题分解”、“清晰编码”和“重构”所带来的益处，通过这些手段共同实现高代码可读性，从而便于逻辑扩展并保持高可维护性，因此，其目的并非设计一个用于处理大量编辑内容的复杂通用编辑器。

2. 假定每行的长度都保持在合理的范围内，以便可以直接使用标准的 Python“str”类型来存储每行。行数也保持在合理的数量，这样所有行都可以存储在一个标准的 Python 列表中，并且可以使用标准的列表和字符串操作（如 append、insert、切片、克隆等）来高效地更新这些行。

3. 每个测试用例旨在评估您的程序的功能和正确性，而非其速度、性能和内存使用情况。每个测试用例都包含多个编辑命令，其中“文本”较短，“重复次数”较小。

4. 该文本编辑器只需处理常规的英文字符，因此，如果有的话，额外的 Unicode 支持是不必要的。

启动选项———

不适用

技能

在这次作业中，您将接受以下内容的培训：

- 重构——基于现有逻辑实现逻辑复用或简化。
- 变量作用域：全局、局部和函数形参。
- 编码风格　（命名约定、有意义的名称、注释、文档字符串等等）
- 问题分解、代码整洁、自顶向下设计
- 用于程序结构和逻辑分解的函数（含参数和返回值）
- 标准对象（字符串、数字和列表）
- 变量作用域

允许的模块

仅允许使用以下 Python 模块:

- 正则表达式（re）

可交付成果

程序源代码（A3_SSE_学号.py），其中 School 为 SSE、SDS、SME、HSS、FE、LHS、MED，StudentID 为您的 9 位学号。
例如，来自中小企业学院的学生 ID 为“119010001”的学生将把 Python 课程命名为:

- A3_SME_119010001.py:

请确保您的源文件以标准的常规 UTF-8 编码格式保存。在 Visual Studio Code 的状态栏中，您可以按如下方式查看当前的编码格式:

在某些情况下，编码方案设置为带有字节顺序标记（BOM）的 UTF-8，如下所示:

字节顺序标记（BOM）的存在可能是由于从网站复制、使用旧版本的编辑器、从其他来源转换、默认编码设置等原因造成的。

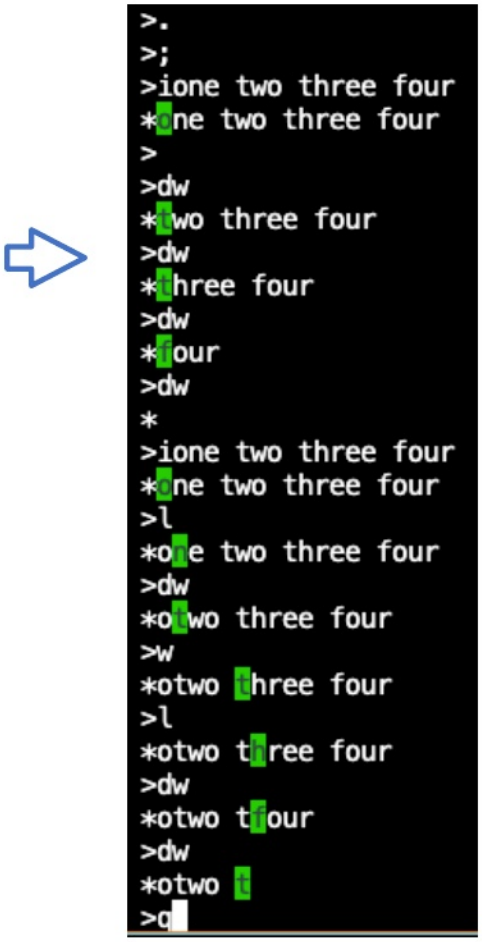
请确认编码格式为 UTF-8 且文件名正确，然后将纯程序文件提交到相应的作业文件夹。如果文件命名错误或编码格式不正确，将扣除 5% 的分数。

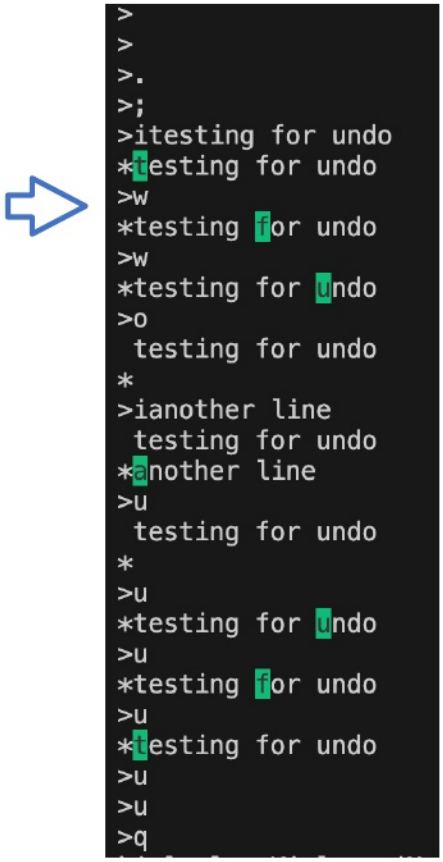
提示与建议

- 在实践中运用课堂上所讲授的问题分解、编写整洁代码和重构的方法。
- 要注意变量的作用域，因为您可能会将一些变量设为全局变量，例如当前编辑器的内容、光标位置、撤销缓冲区等等。
- 有关程序风格和命名约定，请参考 Python 网站（PEP 8）

示例输出

注意：请同时参考第一次作业以获取更多示例输出。





评分标准——

- 编码风格——包括整体程序结构、布局、注释、空格、命名约定、变量、缩进、带有适当参数和返回值的函数。
- 程序正确性——程序是否 100% 符合范围要求运行。
- 用户交互——您的程序与玩家之间信息交换的详尽程度和准确性如何。
- 可读性很重要——结构良好、易于理解的程序，通过使用函数将复杂问题分解为更小、更清晰、更通用的函数，要比一个函数包含复杂逻辑和许多嵌套条件与分支的程序更受青睐！换句话说，对于课程目标而言，具有清晰架构和高可读性的设计比效率更受重视。每个函数中的逻辑应保持简单和简短，并且应设计为执行单一任务，并根据需要通过参数实现通用化。
- KISS 方法——保持简单直接。
- 平衡方法 - 您无需提出一个非常优化的解决方案。但是，要在可读性和效率之间取得平衡，并合理使用程序结构。

物品	百分比	备注
编码风格	百分之三十到四十	如果程序无法运行，则得分为 0%
功能	百分之六十到七十	参照范围

到期日

2025 年 5 月 5 日，晚上 11 点 59 分