

## Text Mode Layer (tml) - un módulo de Python para HP Prime para simular un terminal de fuente de ancho fijo, versión 1.00

por Piotr Kowalewski (komame), agosto-octubre de 2024

### ¿Qué es tml y cómo funciona?

El módulo *tml* (Text Mode Layer) sirve como un reemplazo conveniente para el terminal integrado en HP Prime para Python, ofreciendo además características adicionales para la gestión del terminal.

El módulo utiliza el modo gráfico para simular un terminal utilizando fuentes de ancho fijo. Dado que el terminal integrado se restaura una vez que el programa de Python finaliza su ejecución, no es posible utilizar *tml* directamente desde la línea de comandos. Por lo tanto, debe ser importado en un programa y solo puede funcionar hasta que el programa termine. Por esta razón, *tml* incluye su propio manejo de impresión de texto, desplazamiento de pantalla y mecanismos de entrada de usuario. El manejo de la entrada del usuario ha sido implementado de tal manera que se comporta de manera similar a como funciona en una PC, permitiendo que los datos se ingresen en cualquier posición en la pantalla, en lugar de estar restringidos a un campo de entrada dedicado en la parte inferior de la pantalla (que es una limitación del terminal integrado en HP Prime).

Se puede mostrar una barra de estado en la parte inferior del terminal (activada por defecto), la cual puede mostrar cualquier mensaje de texto. La barra de estado también muestra indicadores de teclado (Shift/Alpha).

El módulo *tml* permite el uso de diversas fuentes de mapa de bits monoespaciadas (diferentes estilos y tamaños), pero solo una puede ser usada para una única instancia de *tml*. Después de cargar una fuente, *tml* inicializa un terminal con el número de filas y columnas que el tamaño de fuente seleccionado permite. El ancho de fuente más grande soportado es de 35 píxeles, y el más pequeño es de 4 píxeles, permitiendo un terminal de 9 a 80 caracteres por fila, respectivamente.

Es posible crear fuentes personalizadas y definir tus propios mapeos de teclas para caracteres específicos (símbolos), permitiendo la entrada directa de símbolos desde el teclado, así como su correcta visualización, como por ejemplo para caracteres diacríticos. Consulta la sección 'Características avanzadas' para más detalles.

### ¿Cómo empezar con tml?

Si estás utilizando la aplicación Python integrada o creando tu propia aplicación basada en la aplicación Python, copia el archivo `tml.py` y una de las fuentes seleccionadas (de la carpeta `Fonts`) a tu aplicación. Luego, en el archivo principal de tu aplicación (usualmente `main.py`), importa el módulo *tml*, lo que te permitirá inicializar el terminal.

También es posible utilizar *tml* a través del envoltorio PPL, pero con este enfoque, es crucial que todas las llamadas a procedimientos PPL se realicen desde Python, no al revés.

### Inicialización del Terminal

La forma más simple de inicializar el terminal es simplemente crear una instancia de *tml* sin proporcionar ningún argumento:

---

```
import tml
t = tml.tml()
```

---

Este enfoque inicializa *tml* con los siguientes valores predeterminados:

- usando la primera fuente encontrada (detalles abajo)
- barra de estado habilitada, sin contenido (")
- modo oscuro desactivado
- tamaño de tabulación: 4 caracteres
- mapeo de caracteres extendidos: vacío (consulte la sección 'Características avanzadas')
- mapeo de teclas de símbolos: vacío (consulte la sección 'Características avanzadas')
- buffer de fuente: 9 (G9)

De lo anterior, se puede ver que *tml* tiene 7 argumentos, cada uno con su propio valor predeterminado; por lo tanto, se recomienda referenciarlos por nombre.

El constructor completo es el siguiente:

---

```
import tml
t = tml.tml(font, status = '', dark_mode = False, tab_size = 4, ext_char_map = {},
            symb_key_map = {}, grob = 9)
```

---

## Argumentos:

### - font:

Durante la inicialización, *tml* busca automáticamente y carga un archivo de fuente (un archivo con extensión '.font'). Sin embargo, también es posible especificar explícitamente la fuente que se cargará usando el parámetro de fuente (cuando se especifica el nombre del archivo de fuente explícitamente, omite la extensión '.font'). Si no se especifica la fuente y la aplicación contiene más de un archivo de fuente, se cargará el primero en orden alfabético.

El paquete del módulo *tml* incluye varias fuentes de muestra en diferentes estilos y tamaños:

- atari8x8 (40 columnas, 28 o 30 filas) - estilo atari de 8 bits
- std5x10 (64 columnas, 26 o 24 filas) - fuente estándar
- std5x12d (64 columnas, 18 o 20 filas) - fuente estándar con caracteres diacríticos
- med6x12 (53 columnas, 18 o 20 filas) - fuente mediana
- med10x12d (32 columnas, 18 o 20 filas) - fuente mediana con caracteres diacríticos
- mini4x7 (80 columnas, 32 o 34 filas) - estilo de fuente mini hp48/49/50

También se pueden utilizar fuentes personalizadas, como se describe en la sección 'Características avanzadas'.

### - status:

Acepta un valor de texto para mostrar en la barra de estado. Por defecto, es un valor vacío ("), lo que causa que la barra de estado aparezca pero no contenga contenido inicial. Introducir cualquier texto en este parámetro hará que aparezca en la barra de estado, que se encuentra en la parte inferior de la pantalla. Cuando la barra de estado es visible, ocupa las últimas dos filas del terminal y no es desplazable. Para desactivar la barra de estado y hacer que el terminal sea de pantalla completa, establezca el parámetro de estado a `None` durante la inicialización. Sin embargo, tenga en cuenta que esto también ocultará los indicadores de estado del teclado (estados de las teclas Shift y Alpha).

### - dark\_mode:

Por defecto, el modo oscuro está desactivado, lo que significa que el fondo del terminal es blanco y los caracteres mostrados son oscuros. Cuando el modo oscuro está activado, el fondo se vuelve negro y los caracteres son claros. Para activar el modo oscuro, establezca el argumento en `True`. Este parámetro solo puede configurarse durante la inicialización del terminal y no se puede cambiar más tarde.

### - **tab\_size:**

El tamaño de tabulación permite dividir la pantalla en segmentos verticales de un tamaño especificado, a los cuales se moverá el cursor cuando se imprima un carácter de tabulación (`\t`). Esto facilita la alineación del texto o la división de columnas de manera más sencilla.

### - **grob:**

Dado que la fuente se carga desde un archivo `.png`, debe almacenarse en búfer en uno de los objetos gráficos de HP Prime (variables gráficas). Por defecto, esto es 9 (G9), que se puede especificar durante la inicialización usando el argumento `grob`. Al igual que `dark_mode`, el valor de `grob` solo se puede establecer durante la inicialización del terminal y no se puede cambiar más tarde.

Ejemplo de inicialización de un terminal con 53 columnas y 20 filas en modo oscuro, con la barra de estado configurada en `'Hello World!'` y mostrando el texto `'This\n\tis\n\t\ta test'` usando `print`, que interpretará los caracteres de nueva línea y tabulación:

---

```
import tml

t = tml.tml(status='Hello World!', font='med6x12', dark_mode=True)
t.print('This\n\tis\n\t\ta test')
t.print('\n\nPress any key...')
t.read_key()
```

---

Efecto obtenido en la pantalla:



## Métodos disponibles

El módulo `tml` proporciona los siguientes métodos:

### - **print(\*args)**

Permite la salida de texto y otros valores al terminal. Acepta argumentos y funciona como el comando `print` integrado en MicroPython en HP Prime, interpretando además el carácter de tabulación (`\t`).

### - **clear()**

Limpia la pantalla del terminal y coloca el cursor en la posición (0,0), que es la esquina superior izquierda de la pantalla. No requiere argumentos.

### - **set\_cursor(x, y)**

Establece el cursor en la posición (x, y). El siguiente uso de *print* o *input* comenzará desde esta posición.

### - **input(prompt, length, alpha, shift, new\_line)**

La función de entrada permite la entrada de datos por parte del usuario, similar a la función de entrada integrada, pero con algunas extensiones que permiten acciones adicionales durante la entrada de datos. Cuando aparece el campo de edición, el usuario puede mover el cursor hacia la izquierda y derecha dentro del texto ingresado, insertar caracteres y eliminarlos usando la tecla Backspace o Del. Durante la entrada de datos, las teclas modificadoras Shift y Alpha (y sus estados de bloqueo) pueden usarse de acuerdo con el comportamiento estándar en la Vista de Inicio. Cuando se activa cualquier tecla modificadora, el estado del teclado se muestra en la barra de estado (si está habilitada), donde 'SL' representa Shift-Lock, 'AL' o 'al' indica Alpha-Lock dependiendo del caso, y '^' indica Shift activo (para un solo uso).

También es posible ingresar caracteres personalizados manteniendo presionada la tecla Symb mientras se presionan otras teclas. Sin embargo, esto requiere definir mapeos de teclas adicionales, como se menciona en la sección 'Características avanzadas'.

Si se presionan Esc o Clear (Shift+Esc) durante la edición, el campo de edición se limpiará y el cursor volverá a la posición inicial. Presionar Enter termina la entrada y devuelve los datos ingresados como una cadena.

*Nota: La entrada de teclado puede no funcionar correctamente en el Virtual Calculator al presionar teclas de letras o símbolos. Si desea probar el método de entrada, use exclusivamente teclas mapeadas (consulte la sección de mapeo del teclado de la computadora en la ayuda de VC para obtener detalles) o use un HP Prime físico.*

*Input* acepta los siguientes argumentos:

- **prompt:** Especifica el mensaje que aparecerá en la pantalla en el campo de entrada de datos.
- **length:** Por defecto, *input* crea un campo de edición que abarca toda la línea (desde la posición x del cursor hasta el borde derecho del terminal). Sin embargo, se puede especificar una longitud máxima del campo de edición, evitando que se ingresen más caracteres de los indicados por el argumento de longitud.
- **alpha:** Habilita inicialmente el Alpha-Lock cuando aparece el campo de edición, permitiendo que el texto se ingrese inmediatamente sin necesidad de presionar la tecla Alpha.
- **shift:** Habilita inicialmente Shift o Shift-Lock (si Alpha-Lock fue activado previamente).
- **new\_line:** Este argumento especifica si presionar Enter debería resultar en moverse a una nueva línea. Establecer *new\_line* en Falso es útil cuando se recopilan datos en la última línea del terminal y no se desea que la pantalla se desplace después de presionar alguna de estas teclas.

### - **read\_key(code=False)**

Pausa el programa y espera a que se presione cualquier tecla (excepto Alpha y Shift). Por defecto, devuelve el símbolo de la tecla presionada, teniendo en cuenta los estados actuales de Alpha y Shift. Si se establece *code* en *True*, devuelve el código de la tecla presionada (0-51).

### - **get\_keys()**

Devuelve una lista de códigos de las teclas actualmente presionadas (incluyendo Alpha y Shift, así como cuando se presionan varias teclas simultáneamente). Devuelve una lista vacía cuando no se presiona ninguna tecla.

### - `set_status(text)`

Permite establecer el texto que se muestra en la barra de estado. Para limpiar la barra de estado, usa una cadena vacía: `set_status('')`

### - `print_xy(x, y, text)`

Muestra el texto especificado en la posición (x, y) del terminal sin mover el cursor. Para esta función, los caracteres de tabulación y nueva línea no son interpretados.

## Características avanzadas

Esta sección está destinada a usuarios avanzados y describe tres características de *tml*:

1. Creación de archivos de fuentes personalizadas.
2. Mapeo de símbolos gráficos de un archivo de fuente a caracteres individuales mostrables.
3. Mapeo de teclas del teclado para la entrada directa de símbolos personalizados.

También encontrarás información aquí sobre cómo funciona *tml* a un nivel más bajo.

Cada archivo de fuente de *tml* es un archivo `.png`, pero ligeramente modificado, ya que debe contener información adicional sobre el ancho de la fuente y el número de configuración de la fuente específica. Internamente, un archivo `.png` que contiene la apariencia de los caracteres individuales de la fuente se trata como un arreglo de 0 a 94, donde 0 es un espacio y 94 es una tilde. Todos los caracteres están organizados lado a lado en una sola fila con un ancho de píxel fijo, por lo que al mostrar caracteres usando el método *print*, cada carácter puede ser mapeado al índice apropiado y la sección correspondiente del archivo de fuente puede ser mostrada para representar la apariencia del carácter específico.

Por defecto, solo los caracteres dentro del rango de la tabla ASCII básica pueden ser mostrados e ingresados en *tml*, es decir, desde el carácter de espacio (código 32) hasta la tilde (código 127). Todos los caracteres en este rango son manejados por el mapeo integrado, por lo que no es necesario definir mapeos adicionales para mostrar estos caracteres o mapeos de teclas para ingresarlos. Sin embargo, es posible usar archivos de fuente con un conjunto más amplio de caracteres. Esto requiere el uso de dos argumentos adicionales al inicializar *tml*: `ext_char_map` y `symb_key_map`.

### Creación de archivos de fuentes personalizadas

Para crear una fuente personalizada, crea un mapa de bits con dimensiones de `x=95*ancho_del_carácter` y `y=altura_del_carácter`. El mapa de bits debe incluir al menos los caracteres ASCII desde el código 32 (espacio) hasta el 126 (tilde) en una sola fila. Los caracteres deben ser negros sobre un fondo blanco. Se permiten tonos de escala de grises, pero se deben evitar otros colores, ya que pueden causar artefactos durante la edición de texto en la pantalla o cuando el modo oscuro está activado.

Si deseas que tu fuente incluya más de 95 caracteres (por ejemplo, si quieres agregar marcas diacríticas o símbolos dedicados necesarios en tu programa), puedes agregarlos en cualquier orden después del carácter tilde, lo que significa que el índice del primer carácter adicional será 95 (como recordatorio: la tilde es el carácter 95, pero dado que comenzamos a contar desde cero, el índice de la tilde es 94). Una vez que todos los caracteres estén diseñados, guarda el archivo de mapa de bits en formato `.png` (puedes establecer la profundidad de color a 1-bit si solo se usaron colores blanco y negro).

Para que el archivo sea reconocido e importado correctamente por el módulo *tml*, se debe agregar un byte de configuración al final del archivo. Este byte debe incluir el número de configuración y el ancho de un solo carácter en píxeles. El número de configuración debe almacenarse en los tres bits

menos significativos de este byte y actualmente siempre debe estar configurado en 0 (otros números de configuración están reservados para uso futuro), mientras que el ancho de un solo carácter debe almacenarse en los cinco bits más significativos como un valor disminuido en 4. Esto permite anchos de fuente que van de 4 a 35 píxeles.

Por ejemplo, si el archivo de fuente contiene caracteres con un ancho de 9 píxeles, el byte de configuración debería verse así en binario: 00101000, que es 40 en decimal. Puedes agregar este byte en una PC o directamente en HP Prime usando el comando AFilesB.

Una vez que se agrega este byte y se cambia la extensión del archivo a '.font', el archivo de fuente está listo para ser utilizado en *tml*.

## Mapeo de símbolos gráficos de un archivo de fuente a caracteres individuales mostrables

Si tu fuente contiene más de los 95 caracteres estándar, necesitas definir un mapeo de caracteres Unicode a los índices correspondientes en tu fuente para que *tml* sepa qué caracteres mostrar para los símbolos adicionales fuera del rango ASCII estándar.

Por ejemplo, supongamos que el carácter 95 es el símbolo de párrafo (§), que no es parte del conjunto de caracteres ASCII estándar. Si el método *print* encuentra este carácter en el texto a imprimir, necesita saber su posición dentro de tu fuente, es decir, necesita conocer su índice. Para mapear el carácter a un índice, necesitas definir un diccionario en la siguiente forma: { '§' : 95 } y pasarlo como el argumento *ext\_char\_map*.

Por ejemplo, supongamos que tu fuente incluye símbolos para los palos de cartas:

♠ (Picas), ♥ (Corazones), ♦ (Diamantes), ♣ (Tréboles), respectivamente desde el índice 95 hasta el 98.

En este caso, el diccionario definido en *ext\_char\_map* debería verse así:

```
{ '♠' : 95, '♥' : 96, '♦' : 97, '♣' : 98 }
```

Si has diseñado símbolos personalizados que no tienen caracteres Unicode correspondientes, puedes usar cualquier carácter Unicode que no necesites en tu programa y reemplazarlo con tu símbolo.

No es necesario definir todos los caracteres diseñados en el diccionario. Sin embargo, si un carácter que ha sido omitido aparece en el texto proporcionado al método *print*, simplemente no se mostrará en la pantalla.

## Mapeo de teclas del teclado para la entrada directa de símbolos personalizados

Cuando se ha definido un mapeo para caracteres no estándar y se pueden mostrar en el terminal usando *print*, a veces puede ser necesario ingresarlos desde el teclado. En la Vista de Inicio, para ingresar un carácter específico de un idioma, como 'ä', puedes abrir la herramienta 'Chars' (Caracteres), localizar el símbolo e insertarlo en el campo de edición. Sin embargo, esta herramienta no es accesible cuando se ejecuta un programa que utiliza el módulo *tml*. En tales casos, el método de entrada proporciona una manera de ingresar ciertos caracteres definidos en tu fuente a través de combinaciones de teclas: [Symb]+[cualquier tecla]. Similar al mapeo de fuente a carácter, es necesario definir los mapeos de teclas y pasarlos como el argumento *ymb\_key\_map*.

Este método también permite definir múltiples símbolos bajo una tecla, lo cual puede ser útil cuando se trata con un grupo de símbolos similares, evitando la necesidad de asignar diferentes teclas para cada uno. Por ejemplo, en francés, la letra 'e' tiene cuatro variantes diacríticas: é, è, ê, ë. En tal caso, es mejor colocar las cuatro bajo la combinación [Symb]+[e], lo que será intuitivo y conveniente al ingresar texto que contenga cualquiera de estos caracteres.

Para ingresar cualquiera de estos caracteres en el campo de edición, presiona y mantén presionada la tecla [Symb], luego presiona adicionalmente la tecla [e]. El primer símbolo aparecerá en

la posición del cursor, pero el cursor no se moverá a la siguiente posición hasta que sueltes la tecla [Symb]. En este punto, presionar repetidamente la tecla [e] recorrerá los caracteres definidos bajo esta tecla, y soltar la tecla [Symb] confirmará el carácter actualmente mostrado.

El mapeo de teclas a símbolos implica usar el número de la tecla (que varía de 0 a 51) como una clave en el diccionario. El valor correspondiente debe ser una lista de cuatro elementos, donde cada elemento es una lista de símbolos o `None`. Cada uno de estos cuatro elementos representa un estado diferente del teclado, ya que los símbolos mostrados pueden variar dependiendo de la activación de las teclas Shift, Alpha o una combinación de ambas. Por lo tanto, el valor para cada tecla en el diccionario debe ser una lista de cuatro casos en el siguiente orden:

1. cuando ningún modificador está activo,
2. cuando solo Alpha está activo,
3. cuando solo Shift está activo,
4. cuando ambos están activos.

Por ejemplo, para la tecla con la letra 'e' y los caracteres diacríticos franceses, podrías crear el siguiente diccionario:

```
{ 18: [None, ['è', 'é', 'ê', 'ë'], None, ['È', 'É', 'Ê', 'Ë']] }
```

Esto se debe entender de la siguiente manera: cuando ni Alpha ni Shift están activos, presionar [Symb]+[e] no mostrará ningún símbolo (ya que Alpha no está activo, no hay razón para que aparezcan letras). Sin embargo, si el modificador Alpha está habilitado, presionar [Symb]+[e] mostrará el carácter 'è' (minúscula), y cada presión subsiguiente de [e] recorrerá los símbolos en la lista hasta que se suelte la tecla [Symb]. El tercer valor también es `None`, ya que no queremos mostrar símbolos de letras cuando solo Shift está activo (este podría ser un buen lugar para mostrar algunos símbolos no alfanuméricos únicos en su lugar). El cuarto valor indica qué debe aparecer cuando tanto Alpha como Shift están activos, por lo que en este caso, los símbolos È, É, Ê, Ë (mayúsculas) serán mostrados.

## Problemas conocidos y limitaciones

- En este momento, no es posible restaurar el contenido de la pantalla del terminal después de que ha sido sobrescrito por código externo. Esto también significa que si se crean dos instancias paralelas de *tml*, no es posible alternar entre ellas.
- No hay soporte para mostrar texto y colores de fondo en diferentes colores (esto se debe a las limitaciones de la biblioteca *hprime*).
- Tampoco es posible leer el código de carácter desde la pantalla *tml* en coordenadas específicas.

Si encuentras algún error, tienes una sugerencia interesante o simplemente quieres contactarme, envía un mensaje privado al usuario **komame** en [www.hpmuseum.org/forum/](http://www.hpmuseum.org/forum/).