

Text Mode Layer (tml) - un module Python pour HP Prime simulant un terminal avec police à largeur fixe, version 1.00
par Piotr Kowalewski (komame), août-octobre 2024

Qu'est-ce que tml et comment fonctionne-t-il ?

Le module *tml* (Text Mode Layer) sert de remplacement pratique pour le terminal intégré du HP Prime pour Python, tout en offrant des fonctionnalités supplémentaires pour la gestion du terminal.

Le module utilise le mode graphique pour simuler un terminal avec des polices à largeur fixe. Étant donné que le terminal intégré est restauré dès que le programme Python termine son exécution, il n'est pas possible d'utiliser *tml* directement depuis la ligne de commande. Il doit donc être importé dans un programme et ne peut fonctionner que jusqu'à la fin de l'exécution de celui-ci. Pour cette raison, *tml* inclut sa propre gestion de l'affichage de texte, du défilement de l'écran et des mécanismes de saisie de l'utilisateur. La gestion de la saisie utilisateur a été implémentée de manière à se comporter de façon similaire à celle d'un PC, permettant de saisir des données à n'importe quelle position sur l'écran, au lieu d'être restreint à un champ de saisie dédié en bas de l'écran (ce qui est une limitation du terminal intégré du HP Prime).

Une barre d'état peut être affichée en bas du terminal (activée par défaut), qui peut afficher n'importe quel message texte. La barre d'état affiche également les indicateurs du clavier (Shift/Alpha).

Le module *tml* permet l'utilisation de diverses polices bitmap monospace (différents styles et tailles), mais une seule peut être utilisée pour une instance de *tml*. Après avoir chargé une police, *tml* initialise un terminal avec le nombre de lignes et de colonnes que la taille de la police sélectionnée permet. La largeur maximale de police prise en charge est de 35 pixels, et la plus petite est de 4 pixels, permettant ainsi un terminal avec respectivement 9 à 80 caractères par ligne.

Il est possible de créer des polices personnalisées et de définir vos propres mappages de touches pour des caractères spécifiques (symboles), permettant ainsi une saisie directe de symboles depuis le clavier, ainsi que leur affichage correct, comme pour les caractères diacritiques. Référez-vous à la section 'Fonctionnalités avancées' pour plus de détails.

Comment démarrer avec tml ?

Si vous utilisez l'application Python intégrée ou si vous créez votre propre application basée sur celle-ci, copiez le fichier *tml.py* ainsi qu'une des polices sélectionnées (depuis le dossier *Fonts*) dans votre application. Ensuite, dans le fichier principal de votre application (généralement *main.py*), importez le module *tml*, ce qui vous permettra d'initialiser le terminal.

Il est également possible d'utiliser *tml* via le wrapper PPL, mais dans ce cas, il est essentiel que tous les appels de procédures PPL soient effectués depuis Python, et non l'inverse.

Initialisation du terminal

La façon la plus simple d'initialiser le terminal est de créer une instance de *tml* sans fournir d'arguments :

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

Cette méthode initialise *tml* avec les valeurs par défaut suivantes :

- utilisation de la première police trouvée (détails ci-dessous)
- barre d'état activée, sans contenu (")
- mode sombre désactivé
- taille des tabulations : 4 caractères
- mappage de caractères étendu : vide (reportez-vous à la section 'Fonctionnalités avancées')
- mappage des touches de symboles : vide (reportez-vous à la section 'Fonctionnalités avancées')
- tampon de police : 9 (G9)

Comme indiqué ci-dessus, *tml* dispose de 7 arguments, chacun avec sa propre valeur par défaut. Il est donc recommandé de les référencer par leur nom lors de l'initialisation.

Le constructeur complet est le suivant :

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

Arguments:

- font:

Lors de l'initialisation, *tml* recherche automatiquement et charge un fichier de police (fichier avec extension .font). Cependant, il est également possible de spécifier explicitement la police à charger en utilisant le paramètre font (lorsque vous spécifiez le nom de fichier de la police, omettez l'extension .font). Si la police n'est pas spécifiée et que l'application contient plus d'un fichier de police, le premier dans l'ordre alphabétique sera chargé.

Le package du module *tml* comprend plusieurs exemples de polices dans différents styles et tailles :

- atari8x8 (40 colonnes, 28 ou 30 lignes) - style 8 bits Atari
- std5x10 (64 colonnes, 26 ou 24 lignes) - police standard
- std5x12d (64 colonnes, 18 ou 20 lignes) - police standard avec caractères diacritiques
- med6x12 (53 colonnes, 18 ou 20 lignes) - police de taille moyenne
- med10x12d (32 colonnes, 18 ou 20 lignes) - police de taille moyenne avec caractères diacritiques
- mini4x7 (80 colonnes, 32 ou 34 lignes) - style de mini-police hp48/49/50

Il est également possible d'utiliser des polices personnalisées, comme décrit dans la section 'Fonctionnalités avancées'.

- status:

Accepte une valeur textuelle à afficher dans la barre d'état. Par défaut, elle est vide (") ce qui fait apparaître la barre d'état sans contenu initial. Saisir n'importe quel texte dans ce paramètre entraînera son affichage dans la barre d'état, qui se trouve en bas de l'écran. Lorsque la barre d'état est visible, elle occupe les deux dernières lignes du terminal et n'est pas défilable. Pour désactiver la barre d'état et mettre le terminal en plein écran, définissez le paramètre status sur [None](#) lors de l'initialisation. Notez toutefois que cela masquera également les indicateurs de statut du clavier (états des touches Shift et Alpha).

- dark_mode:

Par défaut, le mode sombre est désactivé, ce qui signifie que le fond du terminal est blanc et que les caractères affichés sont sombres. Lorsque le mode sombre est activé, le fond devient noir et les caractères sont clairs. Pour activer le mode sombre, définissez cet argument sur [True](#). Ce paramètre ne peut être défini que lors de l'initialisation du terminal et ne peut pas être modifié ultérieurement.

- tab_size:

La taille de la tabulation permet de diviser l'écran en segments verticaux de la taille spécifiée, vers lesquels le curseur se déplacera lorsqu'un caractère de tabulation (\t) est imprimé. Cela permet d'aligner le texte ou de diviser plus facilement le contenu en colonnes.

- **grob:**

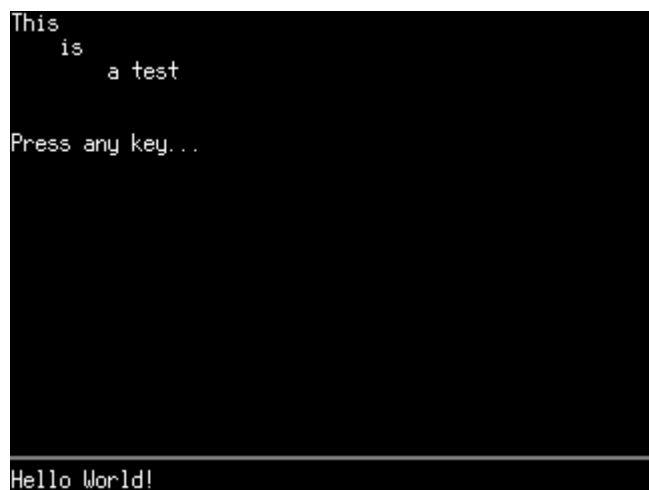
Puisque la police est chargée à partir d'un fichier .png, elle doit être stockée dans l'un des objets graphiques du HP Prime (variables graphiques). Par défaut, il s'agit de 9 (G9), ce qui peut être spécifié lors de l'initialisation à l'aide de l'argument *grob*. Comme le mode *dark_mode*, la valeur de *grob* ne peut être définie que lors de l'initialisation du terminal et ne peut pas être modifiée par la suite.

Exemple d'initialisation d'un terminal avec 53 colonnes et 20 lignes en mode sombre, avec la barre d'état définie sur 'Hello World!' et affichant le texte 'This\n\tis\n\t\tta test' en utilisant la méthode *print*, qui interprétera les caractères de nouvelle ligne et de tabulation :

```
import tml

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

Résultat obtenu à l'écran :



Méthodes disponibles

Le module *tml* propose les méthodes suivantes :

- **print(*args)**

Permet l'affichage de texte et d'autres valeurs sur le terminal. Elle accepte des arguments et fonctionne comme la commande *print* intégrée dans MicroPython sur le HP Prime, tout en interprétant également le caractère de tabulation (`\t`).

- **clear()**

Efface l'écran du terminal et place le curseur à la position (0,0), c'est-à-dire dans le coin supérieur gauche de l'écran. Cette méthode ne prend aucun argument.

- **set_cursor(x, y)**

Place le curseur à la position (x, y). La prochaine utilisation de *print* ou *input* commencera à partir de cette position.

- **input(prompt, length, alpha, shift, new_line)**

La fonction *input* permet la saisie de données par l'utilisateur, similaire à la fonction *input* intégrée, mais avec des extensions qui offrent des actions supplémentaires lors de la saisie de données. Lorsque le champ de saisie apparaît, l'utilisateur peut déplacer le curseur vers la gauche ou la droite dans le texte saisi, insérer des caractères et les supprimer à l'aide des touches Backspace ou Del. Pendant la saisie, les touches de modification Shift et Alpha (ainsi que leurs états de verrouillage) peuvent être utilisées conformément au comportement standard dans la d'accueil. Lorsqu'une touche de modification est activée, l'état du clavier s'affiche dans la barre d'état (si elle est activée), où 'SL' signifie Shift-Lock, 'AL' ou 'al' indique Alpha-Lock selon la casse, et '^' indique un Shift actif (pour une utilisation ponctuelle).

Il est également possible d'entrer des caractères personnalisés en maintenant la touche Symb enfoncée tout en appuyant sur d'autres touches. Cependant, cela nécessite de définir des mappages de touches supplémentaires, comme mentionné dans la section 'Fonctionnalités avancées'.

Si Esc ou Clear (Shift+Esc) est pressé pendant l'édition, le champ de saisie sera effacé et le curseur reviendra à la position de départ. Appuyer sur Enter termine la saisie et renvoie les données saisies sous forme de chaîne de caractères.

Remarque : La saisie au clavier peut ne pas fonctionner correctement sur le Calculateur Virtuel lors de l'appui sur des touches de lettres ou de symboles. Si vous souhaitez tester la méthode input, utilisez exclusivement des touches mappées (reportez-vous à la section Mapping du clavier de l'ordinateur dans l'aide du VC pour plus de détails) ou utilisez un HP Prime physique.

input accepte les arguments suivants :

- **prompt**: Spécifie l'invite qui apparaîtra sur l'écran dans le champ de saisie de données.
- **length**: Par défaut, *input* crée un champ de saisie qui s'étend sur toute la ligne (de la position x du curseur jusqu'au bord droit du terminal). Cependant, il est possible de spécifier une longueur maximale pour le champ de saisie, empêchant ainsi l'utilisateur de saisir plus de caractères que la limite définie par l'argument *length*.
- **alpha**: Active initialement le mode Alpha-Lock lorsque le champ de saisie apparaît, permettant ainsi d'entrer du texte immédiatement sans avoir besoin d'appuyer sur la touche Alpha.
- **shift**: Active initialement le mode Shift ou Shift-Lock (si Alpha-Lock a été précédemment activé).
- **new_line**: Cet argument spécifie si la pression sur Enter doit entraîner le passage à une nouvelle ligne. Définir *new_line* sur *False* est utile lors de la collecte de données sur la dernière ligne du terminal, si vous ne souhaitez pas que l'écran défile après avoir appuyé sur Enter.

- **read_key(code=False)**

Met le programme en pause et attend qu'une touche soit pressée (à l'exception des touches Alpha et Shift). Par défaut, elle renvoie le symbole de la touche appuyée, en tenant compte des états actuels des touches Alpha et Shift. Si *code* est défini sur *True*, elle renvoie le code de la touche appuyée (de 0 à 51).

- **get_keys()**

Renvoie une liste des codes des touches actuellement enfoncées (y compris Alpha et Shift, ainsi que lorsque plusieurs touches sont pressées simultanément). La fonction renvoie une liste vide lorsqu'aucune touche n'est pressée.

- **set_status(text)**

Permet de définir le texte affiché dans la barre d'état. Pour effacer la barre d'état, utilisez une chaîne vide : `set_status('')`

- **print_xy(x, y, text)**

Affiche le texte spécifié à la position (x, y) du terminal sans déplacer le curseur. Pour cette fonction, les caractères de tabulation et de nouvelle ligne ne sont pas interprétés.

Fonctionnalités avancées

Cette section est destinée aux utilisateurs avancés et décrit trois fonctionnalités de *tml* :

1. Création de fichiers de polices personnalisées.
2. Mappage des symboles graphiques d'un fichier de police vers des caractères affichables individuellement.
3. Mappage des touches du clavier pour l'entrée directe de symboles personnalisés.

Vous trouverez également ici des informations sur le fonctionnement de *tml* à un niveau plus bas.

Chaque fichier de police *tml* est un fichier '.png', mais légèrement modifié, car il doit contenir des informations supplémentaires sur la largeur de la police et le numéro de configuration de la police spécifique. En interne, un tel fichier '.png' contenant l'apparence des caractères individuels de la police est traité comme un tableau allant de 0 à 94, où 0 correspond à un espace et 94 à un tilde (~). Tous les caractères sont disposés côte à côte sur une seule ligne avec une largeur de pixel fixe. Ainsi, lors de l'affichage de caractères à l'aide de la méthode *print*, chaque caractère peut être associé à l'index approprié et la section correspondante du fichier de police peut être affichée pour représenter l'apparence de ce caractère spécifique.

Par défaut, seuls les caractères appartenant à la plage de la table ASCII de base peuvent être affichés et saisis dans *tml*, c'est-à-dire du caractère espace (code 32) au tilde (code 126). Tous les caractères de cette plage sont gérés par le mappage intégré, il n'est donc pas nécessaire de définir des mappages supplémentaires pour afficher ces caractères ou des mappages de touches pour les saisir. Cependant, il est possible d'utiliser des fichiers de police avec un ensemble de caractères plus large. Cela nécessite l'utilisation de deux arguments supplémentaires lors de l'initialisation de *tml* : **ext_char_map** et **symp_key_map**.

Création de fichiers de polices personnalisées

Pour créer une police personnalisée, créez une image bitmap avec des dimensions de `x = 95 * largeur_du_caractère` et `y = hauteur_du_caractère`. La bitmap doit inclure au moins les caractères ASCII du code 32 (espace) au code 126 (tilde) sur une seule ligne. Les caractères doivent être noirs sur un fond blanc. Les nuances de gris sont autorisées, mais les autres couleurs doivent être évitées, car elles peuvent provoquer des artefacts lors de l'édition de texte à l'écran ou lorsque le mode sombre est activé.

Si vous souhaitez que votre police inclue plus de 95 caractères (par exemple, pour ajouter des accents diacritiques ou des symboles dédiés nécessaires à votre programme), vous pouvez les ajouter dans n'importe quel ordre après le caractère tilde. Cela signifie que l'index du premier caractère supplémentaire sera 95 (rappel : le tilde est le 95^e caractère, mais comme nous commençons le comptage à zéro, l'index du tilde est 94). Une fois que tous les caractères sont conçus, enregistrez le fichier bitmap au format '.png' (vous pouvez définir la profondeur de couleur sur 1 bit si seules les couleurs noir et blanc ont été utilisées).

Pour que le fichier soit correctement reconnu et importé par le module *tml*, un octet de configuration doit être ajouté à la fin du fichier. Cet octet doit inclure le numéro de configuration et la largeur d'un seul caractère en pixels. Le numéro de configuration doit être stocké dans les trois bits de poids faible de cet octet et doit actuellement être toujours défini à 0 (les autres numéros de configuration sont réservés pour un usage futur), tandis que la largeur d'un caractère doit être stockée dans les cinq bits de poids fort, comme une valeur diminuée de 4. Cela permet de définir des largeurs de police allant de 4 à 35 pixels.

Par exemple, si le fichier de police contient des caractères avec une largeur de 9 pixels, l'octet de configuration doit ressembler à ceci en binaire : 00101000, ce qui correspond à 40 en décimal. Vous pouvez ajouter cet octet sur un PC ou directement sur le HP Prime en utilisant la commande AFilesB.

Une fois cet octet ajouté et l'extension de fichier modifiée en '.font', le fichier de police est prêt à être utilisé dans *tml*.

Mappage des symboles graphiques d'un fichier de police vers des caractères affichables individuellement

Si votre police contient plus de 95 caractères standard, vous devez définir un mappage des caractères Unicode vers les indices correspondants dans votre police pour que *tml* puisse savoir quels caractères afficher pour les symboles supplémentaires en dehors de la plage standard ASCII.

Par exemple, supposons que le 95^e caractère soit le symbole de paragraphe (§), qui ne fait pas partie de l'ensemble de caractères ASCII standard. Si la méthode *print* rencontre ce caractère dans le texte à afficher, elle doit connaître sa position dans votre police, c'est-à-dire son index. Pour mapper le caractère à un index, vous devez définir un dictionnaire sous la forme suivante : { '§' : 95 } et le passer en tant qu'argument *ext_char_map*.

Par exemple, supposons que votre police inclut des symboles pour les couleurs de cartes :

♠ (Pique), ♥ (Cœur), ♦ (Carreau), ♣ (Trèfle), respectivement aux indices 95 à 98.

Dans ce cas, le dictionnaire défini dans *ext_char_map* devrait ressembler à ceci :

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

Si vous avez conçu des symboles personnalisés qui n'ont pas de caractères Unicode correspondants, vous pouvez utiliser n'importe quel caractère Unicode dont vous n'avez pas besoin dans votre programme et le remplacer par votre symbole.

Il n'est pas nécessaire de définir tous les caractères conçus dans le dictionnaire. Cependant, si un caractère qui a été omis apparaît dans le texte fourni à la méthode *print*, il ne sera tout simplement pas affiché à l'écran.

Mappage des touches du clavier pour l'entrée directe de symboles personnalisés

Lorsqu'un mappage pour des caractères non standards a été défini et que ceux-ci peuvent être affichés sur le terminal à l'aide de la commande *print*, il peut parfois être nécessaire de les saisir depuis le clavier. Dans la vue d'accueil, pour entrer un caractère spécifique à une langue, comme 'ä', vous pouvez ouvrir l'outil 'Chars' (Caractères), localiser le symbole, puis l'insérer dans le champ de saisie. Cependant, cet outil n'est pas accessible lors de l'exécution d'un programme qui utilise le module *tml*. Dans ce cas, la méthode *input* permet de saisir certains caractères définis dans votre police via des combinaisons de touches : [Symb] + [n'importe quelle touche].

Similaire au mappage police-caractère, il est nécessaire de définir le mappage des touches et de le passer en tant qu'argument *symb_key_map*.

Cette méthode permet également de définir plusieurs symboles sous une même touche, ce qui peut être utile lorsqu'il s'agit d'un groupe de symboles similaires, évitant ainsi la nécessité d'attribuer une touche différente pour chaque symbole. Par exemple, en français, la lettre 'e' a quatre variantes

diacritiques : é, è, ê, ë. Dans un tel cas, il est préférable de placer ces quatre variantes sous la combinaison [Symb]+[e], ce qui sera intuitif et pratique pour entrer du texte contenant l'un de ces caractères.

Pour saisir l'un de ces caractères dans le champ de saisie, maintenez enfoncée la touche [Symb], puis appuyez sur la touche [e]. Le premier symbole apparaîtra à la position du curseur, mais le curseur ne se déplacera pas vers la position suivante tant que vous n'aurez pas relâché la touche [Symb]. À ce moment-là, appuyer plusieurs fois sur la touche [e] fera défiler les caractères définis sous cette touche, et relâcher [Symb] confirmera le caractère actuellement affiché.

Le mappage des touches vers les symboles se fait en utilisant le numéro de la touche (allant de 0 à 51) comme clé dans le dictionnaire. La valeur correspondante doit être une liste de quatre éléments, où chaque élément est soit une liste de symboles, soit `None`. Chacun de ces quatre éléments représente un état différent du clavier, car les symboles affichés peuvent varier en fonction de l'activation de la touche Shift, de la touche Alpha ou d'une combinaison des deux. Ainsi, la valeur pour chaque touche dans le dictionnaire doit être une liste de quatre cas dans l'ordre suivant :

1. Lorsque aucune touche de modification n'est active,
2. Lorsque seule la touche Alpha est active,
3. Lorsque seule la touche Shift est active,
4. Lorsque les deux touches (Alpha et Shift) sont actives.

Par exemple, pour la touche contenant la lettre 'e' et les caractères diacritiques français, vous pouvez créer le dictionnaire suivant :

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

Cela doit être compris comme suit : lorsque ni Alpha ni Shift ne sont activés, appuyer sur [Symb]+[e] n'affichera aucun symbole (puisque Alpha n'est pas actif, il n'y a aucune raison d'afficher des lettres). Cependant, si le modificateur Alpha est activé, appuyer sur [Symb]+[e] affichera le caractère 'è' (minuscule), et chaque pression supplémentaire sur [e] fera défiler les symboles de la liste jusqu'à ce que la touche [Symb] soit relâchée. La troisième valeur est également définie sur `None`, car nous ne souhaitons pas afficher de lettres lorsque seule la touche Shift est active (ce pourrait être un bon emplacement pour afficher des symboles non alphanumériques uniques). La quatrième valeur indique ce qui doit apparaître lorsque Alpha et Shift sont tous deux activés. Dans ce cas, les symboles È, É, Ê, Ë (majuscule) seront affichés.

Problèmes connus et limitations

- À ce jour, il n'est pas possible de restaurer le contenu de l'écran du terminal après qu'il a été écrasé par du code externe. Cela signifie également que si deux instances parallèles de *tml* sont créées, il n'est pas possible de basculer entre elles.
- Il n'y a pas de prise en charge pour afficher le texte et le fond dans des couleurs différentes (en raison des limitations de la bibliothèque *hprime*).
- Il n'est également pas possible de lire le code d'un caractère depuis l'écran *tml* à des coordonnées spécifiques.

Si vous rencontrez des bugs, avez une suggestion intéressante ou souhaitez simplement me contacter, envoyez un message privé à l'utilisateur **komame** sur www.hpmuseum.org/forum/.