Starplot : une bibliothèque Python pour générer des cartes du ciel

Starplot est une bibliothèque Python permettant de générer des cartes du ciel très personnalisables. Elle est activement développée depuis juin 2023 par Steve Berardi, ingénieur logiciel et astronome amateur dans la banlieue de San Diego en Californie.

Dans sa version 0.9.1, Stardev permet de générer trois types de cartes astronomiques :

- les **cartes planisphériques** (`Zenith Plot`), qui représentent l'entièreté du ciel visible depuis un point d'observation particulier. Il est alors nécessaire de renseigner le point d'observation sur Terre et la date et l'heure de la carte.
- les **cartes stellaires** (`Map Plot`), qui permettent de représenter une zone restreinte du ciel, mais avec bien plus de détails que sur une carte planisphérique. Il n'est pas nécessaire d'indiquer un point d'observation et une date pour ce type de carte puisque l'horizon n'y est pas représenté.
- les **cartes optiques** (`Optic Plot`), qui représentent ce que l'on verrait à l'oculaire d'une lunette ou d'un télescope, ou dans des jumelles. Ces cartes nécessitent le point d'observation sur Terre et la date et l'heure d'observation.

Les cartes incluent évidemment de nombreuses étoiles, tirées du catalogue hipparcos par défaut (118 218), mais il est possible d'utiliser également le catalogue tycho-1, bien plus complet (1 055 115 étoiles). Selon la magnitude de chaque étoile, on pourra afficher son nom si elle est assez brillante ou sa désignation de Bayer. Les 88 constellations peuvent également être incluses dans la carte (intéressant pour les cartes planisphériques ou les cartes stellaires si le champ est suffisant).

Vous pouvez également ajouter les planètes du système solaire, la Lune, les objets du ciel profond, ainsi que la voie lactée (représentée sous la forme d'une surface uniforme).

Enfin, pour terminer ce rapide tour d'horizon, la carte générée peut être exportée soit au format PNG (format par défaut), soit au format SVG, pour être ensuite retravaillé avec un logiciel comme Inkscape.

Sommaire

- Installation
- Tutoriel
    - Carte de base
    - Ajout d'un thème de couleurs
    - Ajout d'un titre
    - Gestion de l'affichage des étoiles
    - Style du nom de la constellation
    - Ajout de l'étoile T-Lyrae
- Conclusion

Installation

Je vais décrire uniquement l'installation sous GNU/Linux, mais si vous utilisez un autre système, ça ne doit pas être fondamentalement différent.

Histoire de ne pas modifier le système hôte, commençons par créer un dossier de travail, puis par déployer l'environnement virtuel Python dans ce dossier :

```
mkdir test_starplot
cd test_starplot
virtualenv venv
```

puis on active l'environnement virtuel :

```
source venv/bin/activate
```

et enfin on lance l'installation de la bibliothèque starplot en utilisant `pip` :

```
pip install starplot
```

Cela va installer `starplot`, mais également automatiquement toutes les dépendances (merci `pip` !).

La suite se fera dans l'environnement de développement intégré VSCodium, dans lequel on aura installé le plugin Python.

Lançons donc VScodium dans le dossier actuel :

```
codium .
```

Vous pouvez ensuite créer un nouveau fichier Python avec l'entrée de menu `File > New file > Python File`. L'interpréteur Python disponible dans l'environnement virtuel Python installé dans le dossier `venv` est automatiquement sélectionné. Rien à faire d'autre que de coder maintenant !

Tutoriel

Pour ce petit tutoriel, on se donne comme objectif de représenter une carte de la constellation de la Lyre, et d'y repérer l'étoile T-Lyrae, étoile carbonée d'un rouge très sombre, très chouette à observer au télescope.

Carte de base

Commençons par une première carte toute simple représentant la constellation de la Lyre. D'abord le code, et ensuite on décortique :

```
# Code 00
from starplot import MapPlot, Projection
```

```
p = MapPlot(
    projection=Projection.MERCATOR,
    ra_min=18,
    ra_max=19.5,
    dec_min=25,
    dec_max=48,
    resolution=800,
)
p.constellations()
p.constellation_borders()
p.stars()

p.export("starplot_00.png")
```

Allons-y tranquillement et commençons par les imports. Depuis la bibliothèque `starplot`, nous devons importer deux classes d'objet :

- **`MapPlot`** qui est la classe de base qui va gérer toute la construction de la carte
- **`Projection`**, qui va nous permettre de définir un type de projection pour la carte. Il n'y a pas de paramètre par défaut pour le type de projection donc nous devons absolument indiquer lequel utiliser.

Ensuite, nous créons l'objet p qui représente notre carte en instanciant la classe `MapPlot`. Nous passons un certain nombre de paramètres lors de la création de l'objet (voir la documentation de MapPlot):

- **`projection=Projection.MERCATOR`** indique que nous souhaitons une projection de type mercator, adaptée pour représenter des cartes pour des déclinaisons entre -70° et 70°, mais qui distord les objets près des pôles. Les autres types de projections disponibles sont décrits dans la documentation de Projection
- **`ra_min` et `ra_max`** correspondent aux limites de la carte en ascension droite, soit ici entre 18 h et 19 h 30.
- **`dec_min` et `dec_max`** sont les limites de la carte en déclinaison, en degrés par rapport à l'équateur céleste.
- **`resolution=800`** correspond à la taille en pixels de la plus grande dimension de la carte. Étrangement, avec 1200 pixels dans le code, l'image fait 102×1318. Attention, plus la résolution est élevée, plus le texte apparaîtra petit en relatif. Heureusement, il est possible de régler ça plus tard en utilisant les styles.

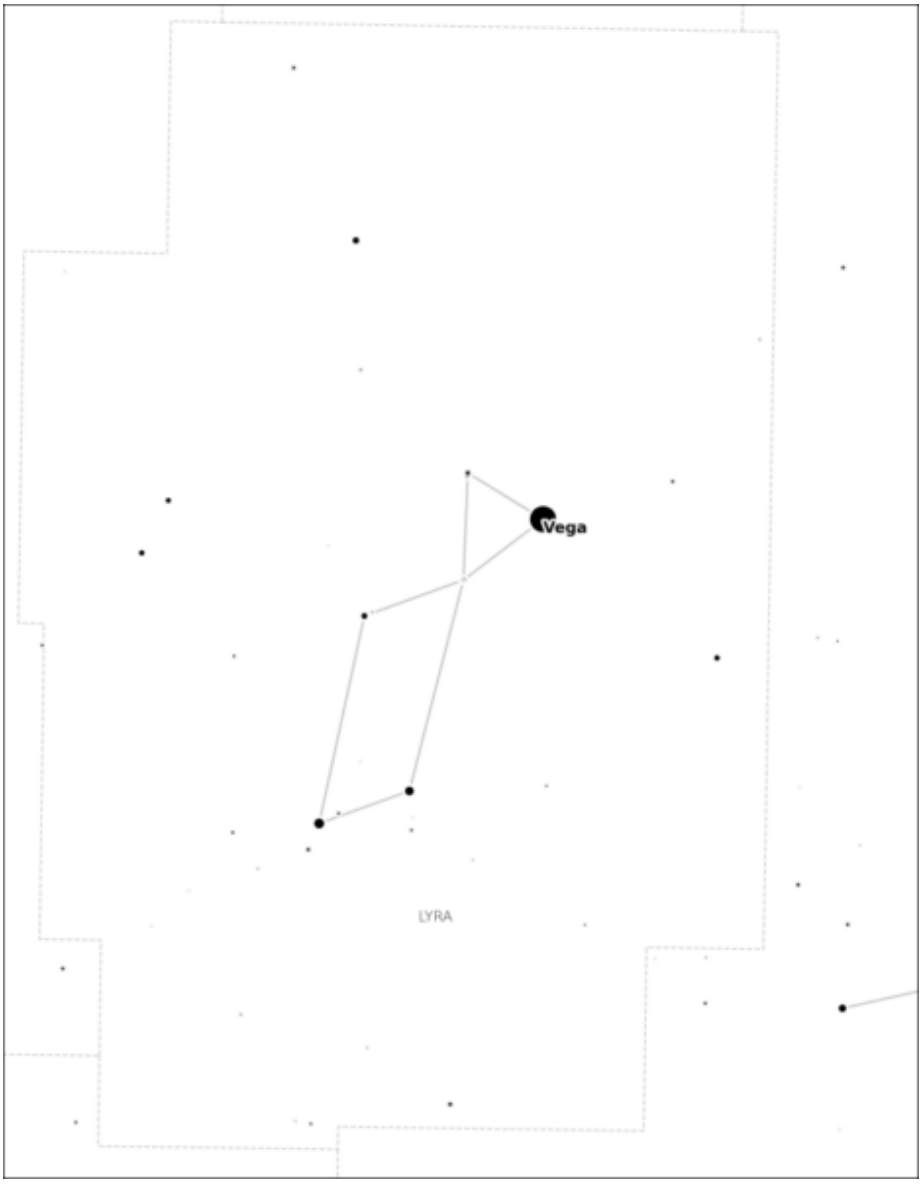
Si on s'arrêtait ici, nous obtiendrions une magnifique carte… totalement vide. Il faut ensuite indiquer ce que l'on souhaite

ajouter sur la carte :

- **`p.constellations()`** permet de relier certaines étoiles par des traits continus pour représenter les constellations. Le nom de la constellation est également ajouté en dessous.
- **`p.constellation_borders()`** dessine les limites des constellations par défauts en traits pointillés
- **`p.stars()`** ajoute les étoiles sur la carte. Par défaut, seules les étoiles de magnitude jusqu'à 6 sont représentées. Certaines étoiles, comme ici Vega, voient leur nom affiché juste à côté.

Cette fois la carte commence à ressembler à quelque chose. Pour la visualiser, il reste à l'exporter dans un fichier avec la commande **`p.export("starplot_00.png")`**. Le format par défaut est PNG, et le format SVG est également possible (voir la documentation de export)

Et voici le rendu :



Carte de la constellation de la Lyre avec les réglages par défaut

Pas très joli, mais ça fait le boulot. Heureusement, on va pouvoir améliorer tout ça avec les styles.

Ajout d'un thème de couleurs

Cette fois on va entrer dans quelque chose de très intéressant avec la personnalisation de la carte. Dans un premier temps, nous allons simplement choisir un thème de couleur. Starplot dispose d'un framework permettant de gérer finement les réglages de la carte.

Pour changer le thème de couleur, il faut suivre les étapes suivantes :

- importation des classes `PlotStyle` et `extensions` depuis la bibliothèque `starplot.styles`
- création d'un objet de type `PlotStyle` vide
- ajout d'extensions prédéfinies au style
- application du style créé à l'objet `MapPlot`

Ce qui donne le code correspondant :

```
# Code 01
from starplot import MapPlot, Projection
from starplot.styles import PlotStyle, extensions

# création de l'objet style
style = PlotStyle()

# ajout d'extensions prédéfinies au style
style = style.extend(
    extensions.BLUE_DARK,
    extensions.MAP,
)


p = MapPlot(
    projection=Projection.MERCATOR,
    ra_min=18,
    ra_max=19.5,
    dec_min=25,
    dec_max=48,
    resolution=800,
    # application du style à l'objet MapPlot
    style=style
)
p.constellations()
```

```
p.constellation_borders()
p.stars()

p.export("starplot_01.png")
```

Il est possible d'ajouter deux types d'extensions prédéfinis au style :

- des thèmes de couleur
- des réglages prédéfinis en fonction du type de carte (Map, Zenith et Optic)

Nous avons choisi ici un thème bleu sombre (`BLUE_DARK`), et les réglages prédéfinis pour les cartes planisphériques (`MAP`). Les autres options sont détaillées dans la documentation des extensions.

Il suffit ensuite d'indiquer quel style utiliser lors de la création de l'objet `p` de type `MapPlot`.

Et voici ce que ça donne :

Carte de la constellation de la Lyre avec un thème de couleur

Ajout d’un titre

La documentation nous indique qu’il existe une méthode title pour ajouter un titre à la carte :

```
title(text: str, style: LabelStyle = None)
```

Cette méthode prend donc deux paramètres : – le titre, obligatoire – un objet de type `LabelStyle`, optionnel et permettant de modifier le style du titre.

En parcourant la documentation de `LabelStyle`, on apprend que cette classe d’objet se trouve dans la bibliothèque `starplot.styles`. Il faudra donc l’importer. On trouve, toujours dans cette documentation, une référence à la classe `ColorStr`, qui permet d’indiquer une couleur au format hexadécimal. Cette classe est également définie dans la bibliothèque `starplot.styles`.

Ajoutons donc un titre à notre carte. Je reporte ici uniquement les changements principaux :

```
# Code 02
[...]
# importation de classes permettant de gérer l'apparence des labels
from starplot.styles import LabelStyle, ColorStr, FontWeightEnum


[...]
p.stars()

# création du style pour le titre
style_titre = LabelStyle(
    font_size=20,
    font_color=ColorStr('#bfe0ff'),
    font_weight=FontWeightEnum.BOLD,
    line_spacing=2,
)

# ajout du titre de la carte et indication du style à utiliser
p.title("Étoile carbonnée\nT-Lyrae", style=style_titre)

p.export("starplot_02.png")
```

Vous noterez au passage la possibilité de revenir à la ligne dans la chaîne de caractère utilisée pour le titre (`\n`).

Autre caractéristique, la graisse de la police est indiquée avec l’énumération`font_weight=FontWeightEnum.BOLD`, mais la documentation indique que l’on aurait pu également utiliser la chaîne de caractère `bold`. Il y a cependant deux intérêts à utiliser l’énumération :

- d’une part, nous sommes sûrs de ne pas nous tromper dans l’orthographe

du mot clé (OK, pour bold, ce n'est pas très compliqué, mais ça peut parfois l'être)
- d'autre part, la complétion automatique de VSCodium permet d'avoir une vision rapide de toutes les options possibles.

Il ne faut cependant pas oublier d'importer la classe `FontWeightEnum` depuis la bibliothèque `starplot.styles`.

Toutes les énumérations disponibles sont indiquées dans la documentation.

Ajout d'un titre à la carte

Après compilation, voici la nouvelle carte avec le titre :

Gestion de l'affichage des étoiles

On souhaiterait maintenant modifier l'apparence des étoiles : – mise en forme des labels des étoiles – affichage de la dénomination de Bayer pour certaines étoiles (et mise en forme par la même occasion) – gestion du symbole pour représenter chaque étoile, et de son diamètre en fonction de la magnitude de l'étoile

D'après la documentation, le style des étoiles se fait via un objet de type `ObjectStyle`, qu'il faudra d'abord importer depuis la bibliothèque `starplot.style`. Cette classe permet de créer des styles pour les différents objets affichés sur la carte, et pas uniquement les étoiles. L'objet `ObjectStyle` se compose de deux paramètres :

- un `label`, de type `LabelStyle`, qui permet de gérer l'apparence du texte du label de l'étoile (type de police de caractère, taille, couleur, gras, italique...). Nous l'avons déjà importée depuis la bibliothèque `starplot.style` pour le titre de la carte.
- un `marker` de type `MarkerStyle`, qui correspond au symbole utilisé pour représenter l'étoile. Par défaut, il s'agit d'un disque noir, mais on peut en modifier l'apparence.

Au niveau des importations, nous aurons donc :

```
from starplot.styles import LabelStyle, ColorStr, FontWeightEnum, ObjectStyle, MarkerStyle, MarkerSymbolEnum, FillStyleEnum
```

J'ai ajouté également deux énumérations : `MarkerSymbolEnum` pour choisir facilement le symbole pour représenter un objet, et `FillStyleEnum` pour choisir comment remplir le symbole.

En début de code, nous avons créé un style pour la carte et avons utilisé les extensions pour paramétrer tout un ensemble de réglages pour le style :

```
style = PlotStyle()

style = style.extend(
    extensions.BLUE_DARK,
    extensions.MAP,
)
```

Si nous voulons apporter des modifications à notre style, il faut le faire **après** l'utilisation des extensions pour que nos changements ne soient pas écrasés par l'application du style prédéfini.

On peut alors voir dans la documentation de `PlotStyle` comment modifier l'apparence du texte de la désignation de Bayer pour les étoiles :

```
style.bayer_labels = LabelStyle(
        font_size=20,
        font_color = 'white',
        font_weight=FontWeightEnum.LIGHT,
        )
```

Vous noterez que l'on peut indiquer directement une couleur ('white') sans passer par un objet de type `ColorStr`.

Intéressons-nous maintenant au tracé des étoiles. Nous créons un style de type `ObjectStyle` comprenant un `LabelStyle` pour les labels des étoiles (hors désignation de Bayer) et un `MarkerStyle` pour le symbole de l'étoile :

```
style_etoiles = ObjectStyle(
    label = LabelStyle(
        font_size = 25,
        font_color = 'white',
    ),
    marker = MarkerStyle(
        symbol = MarkerSymbolEnum.CIRCLE,
        color='white',
        fill=FillStyleEnum.FULL,
    )
)
```

Il ne reste plus qu'à modifier la ligne qui demande à tracer les étoiles dans la carte, en indiquant que l'on souhaite afficher la désignation de Bayer pour les étoiles peu lumineuses et donner le style à appliquer :

```
p.stars(
    style=style_etoiles,
    bayer_labels=True,
)
```

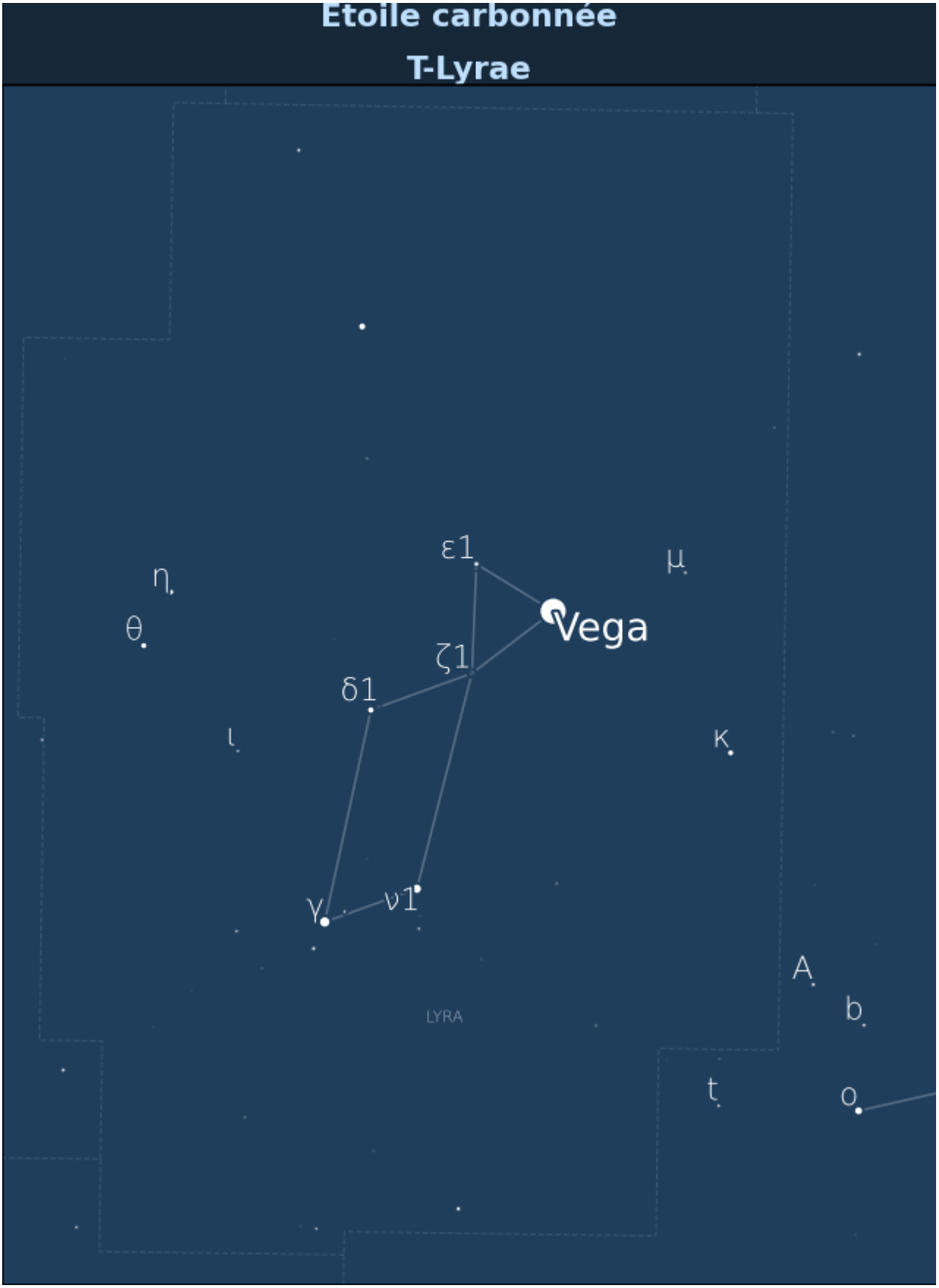
Et voilà, c'est déjà beaucoup mieux. Voici le code complet pour la génération de cette carte :

```
# Code 03
from starplot import MapPlot, Projection
from starplot.styles import PlotStyle, extensions
from starplot.styles import LabelStyle, ColorStr, FontWeightEnum, ObjectStyle, MarkerStyle, MarkerSymbolEnum, FillStyleEnum

style = PlotStyle()

style = style.extend(
    extensions.BLUE_DARK,
    extensions.MAP,
)

# Modification du style de la carte pour les désignations de Bayer
style.bayer_labels = LabelStyle(
        font_size=20,
        font_color = 'white',
        font_weight=FontWeightEnum.LIGHT,
        )
```

```
p = MapPlot(
    projection=Projection.MERCATOR,
    ra_min=18,
    ra_max=19.5,
    dec_min=25,
    dec_max=48,
    resolution=800,
    style=style
)

p.constellations()
p.constellation_borders()

# Création d'un style spécifique pour l'affichage des étoiles
# hors désignation de Bayer
style_etoiles = ObjectStyle(
    label = LabelStyle(
        font_size = 25,
        font_color = 'white',
    ),
    marker = MarkerStyle(
        symbol = MarkerSymbolEnum.CIRCLE,
        color='white',
        fill=FillStyleEnum.FULL,
    )
)

# Application du style pour les étoiles
# et activation de l'affichage de la désignation de Bayer
p.stars(
    style=style_etoiles,
    bayer_labels=True,
)

style_titre = LabelStyle(
    font_size=20,
    font_color=ColorStr('#bfe0ff'),
    font_weight=FontWeightEnum.BOLD,
    line_spacing=2,
)
```

```
p.title("Étoile carbonnée\nT-Lyrae", style=style_titre)

p.export("starplot_03.png")
```

Et voici le rendu de la carte après exécution du code Python. Ça commence à ressembler à quelque chose de sympa. Le nom de la constellation est encore un peu petit, mais nous allons régler ça assez facilement dans la section suivante.

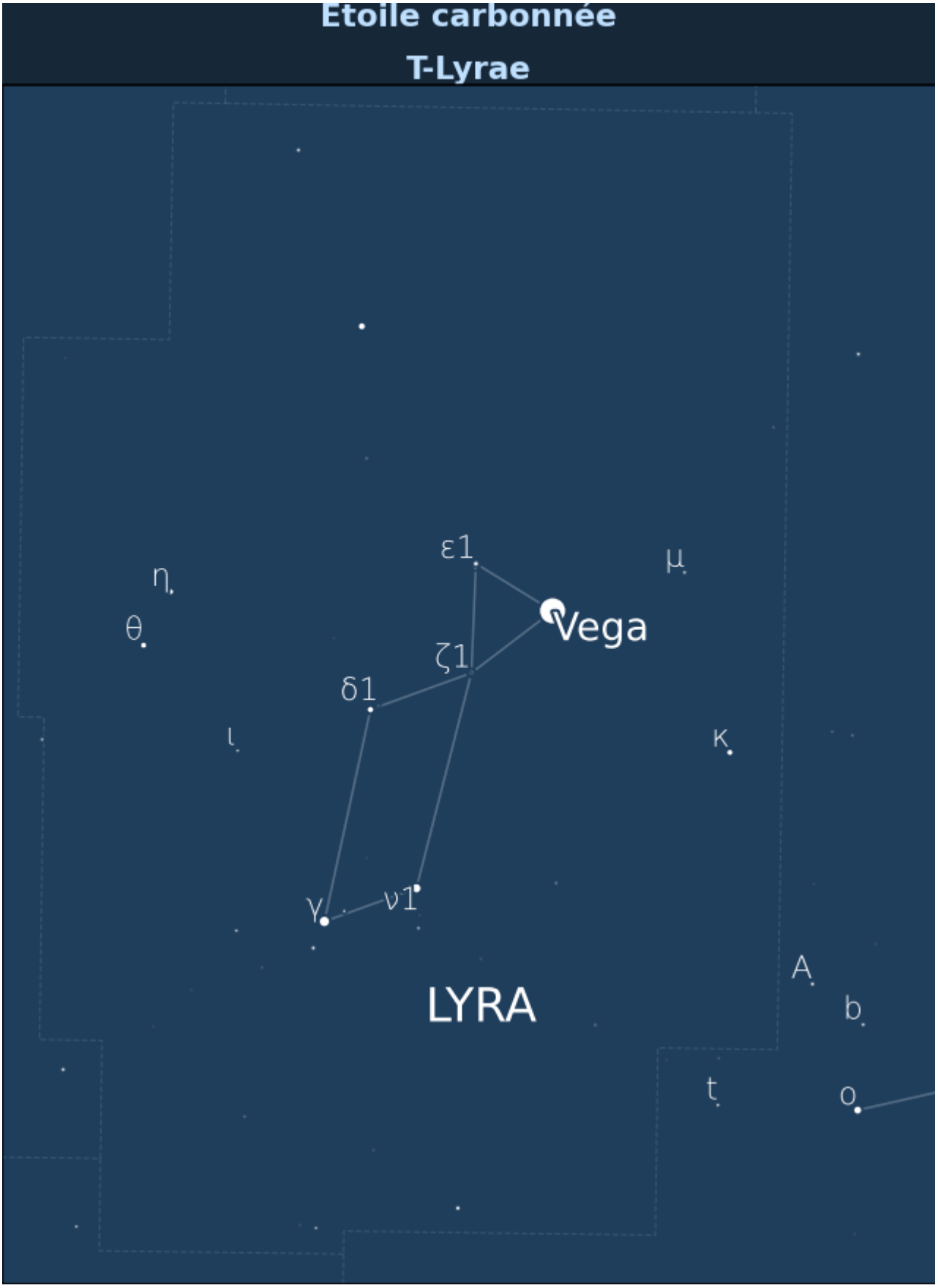


Style du nom de la constellation

La documentation de PlotStyle.constellation nous indique comment modifier facilement l'apparence du label des constellation. Nous ajoutons donc à la suite de la définition de `style.bayer_labels` le code suivant :

```
# style des labels des constellations
style.constellation.label = LabelStyle(
    font_size = 30,
    font_color = 'white',
)
```

Et voilà, le nom de la constellation est maintenant bien visible. Je ne redonne pas ici le code complet, mais juste le résultat :



Ajout de l’étoile T-Lyrae

Nous allons maintenant ajouter **manuellement** un objet sur la carte : l’étoile T-Lyrae. Tout d’abord, il nous faut récupérer les coordonnées de l’étoile dans le ciel, par exemple à partir du site universeguide.com :

- ascension droite : 18 h 32m 20.08 soit 18.54 h
- déclinaison : +36° 59` 55.7 soit 37.00°

Pour ajouter un nouvel objet dans la carte, il faut utiliser la méthode `marker` de l'objet `MapPlot`, comme indiqué dans la documentation.

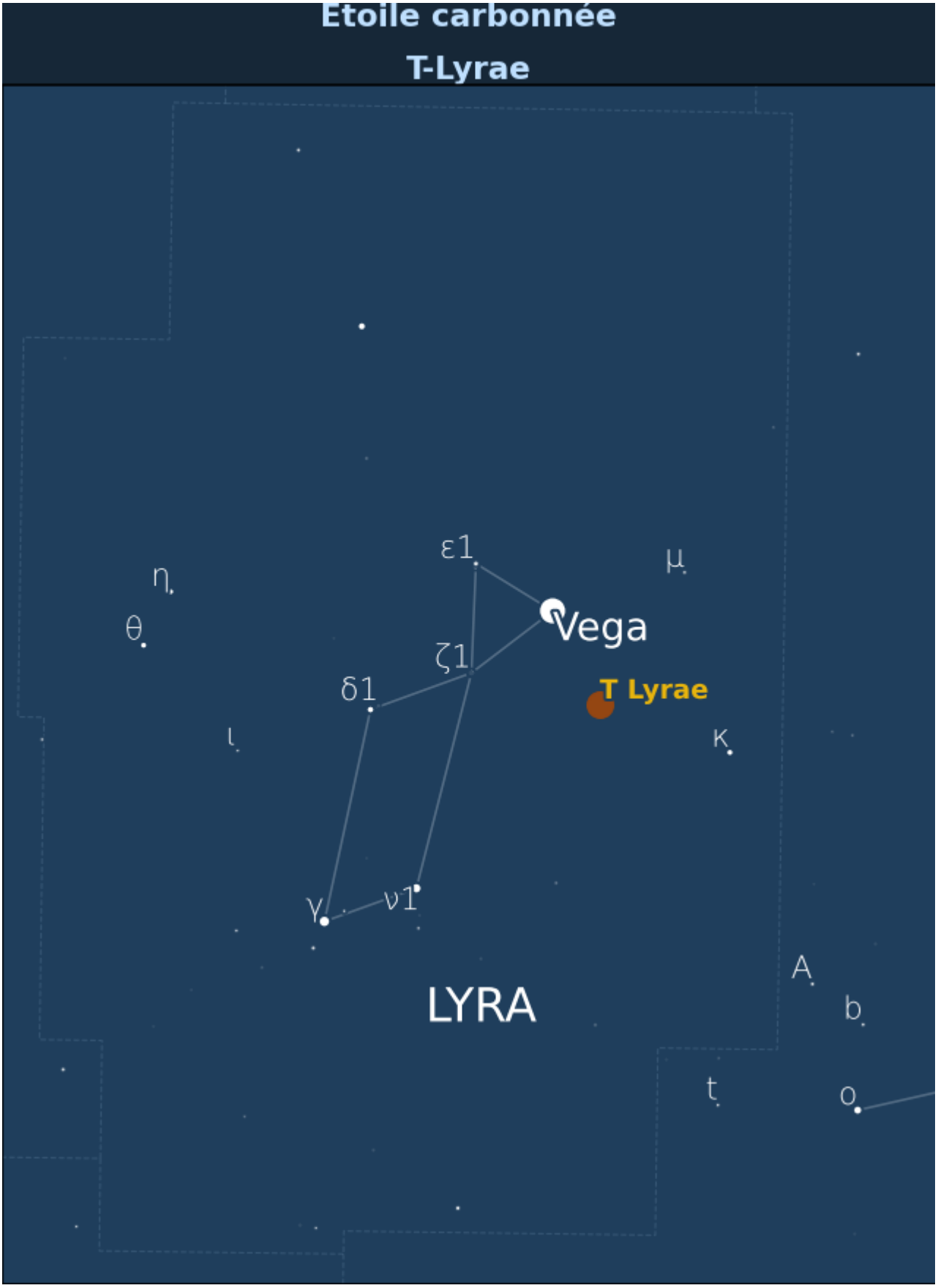
Les paramètres à renseigner sont les suivants :

- ra : ascension droite de l'objet
- dec : déclinaison de l'objet
- label : étiquette à afficher à côté de l'objet
- style : un ensemble de `ObjectStyle` sous forme de dictionnaire, permettant de gérer à la fois l'apparence de la marque, et celle du label
- legend_label : le texte à afficher dans la légende. Nous n'avons pas utilisé la légende de la carte dans ce tuto donc on ne mettra rien (par défaut, ce texte est vide).

Voici le code à ajouter en fin de script Python, avant l'export dans l'image PNG :

```
p.marker(
    ra=18.54,
    dec=37,
    label="T Lyrae",
    style={
        "marker": {
            "size": 16,
            "symbol": "circle",
            "fill": "full",
            "color": "#944612",
            "edge_color": "#944612",
        },
        "label": {
            "font_size": 17,
            "font_weight": "bold",
            "font_color": "#e7b40e",
        },
    },
)
```

Ce qui après exécution du script, nous donne la carte suivante :



Notez que le style défini sous la forme de dictionnaire Python, où les paramètres sont donnés sous forme de chaînes de caractère, peut être également écrit en utilisant uniquement des objets Python :

```
p.marker(
```

```
    ra=18.54,
    dec=37,
    label="T Lyrae",
    style=ObjectStyle(
        marker=MarkerStyle(
            size=16,
            symbol=MarkerSymbolEnum.CIRCLE,
            fill=FillStyleEnum.FULL,
            color=ColorStr('#944612'),
            edge_color=ColorStr('#944612'),
        ),
        label=LabelStyle(
            font_size=17,
            font_weight = FontWeightEnum.BOLD,
            font_color = ColorStr('#e7b40e'),
        )
    )
)
```

Voici donc pour terminer le code dans sa version complète :

```
# Code 05
from starplot import MapPlot, Projection
from starplot.styles import PlotStyle, extensions
from  starplot.styles  import  LabelStyle,  ColorStr,
FontWeightEnum, ObjectStyle, MarkerStyle, MarkerSymbolEnum,
FillStyleEnum

style = PlotStyle()

style = style.extend(
    extensions.BLUE_DARK,
    extensions.MAP,
)

style.bayer_labels = LabelStyle(
        font_size=20,
        font_color = 'white',
        font_weight=FontWeightEnum.LIGHT,
        )

style.constellation.label = LabelStyle(
```

```
    font_size = 30,
    font_color = 'white',
)

p = MapPlot(
    projection=Projection.MERCATOR,
    ra_min=18,
    ra_max=19.5,
    dec_min=25,
    dec_max=48,
    resolution=800,
    style=style
)

p.constellations()
p.constellation_borders()

style_etoiles = ObjectStyle(
    label = LabelStyle(
        font_size = 25,
        font_color = 'white',
    ),
    marker = MarkerStyle(
        symbol = MarkerSymbolEnum.CIRCLE,
        color='white',
        fill=FillStyleEnum.FULL,
    )
)

p.stars(
    style=style_etoiles,
    bayer_labels=True,
)

style_titre = LabelStyle(
    font_size=20,
    font_color=ColorStr('#bfe0ff'),
    font_weight=FontWeightEnum.BOLD,
    line_spacing=2,
)
```

```
p.title("Étoile carbonnée\nT-Lyrae", style=style_titre)

p.marker(
    ra=18.54,
    dec=37,
    label="T Lyrae",
    style=ObjectStyle(
        marker=MarkerStyle(
            size=16,
            symbol=MarkerSymbolEnum.CIRCLE,
            fill=FillStyleEnum.FULL,
            color=ColorStr('#944612'),
            edge_color=ColorStr('#944612'),
        ),
        label=LabelStyle(
            font_size=17,
            font_weight = FontWeightEnum.BOLD,
            font_color = ColorStr('#e7b40e'),
        )
    )
)

p.export("starplot_05.png")
```

Conclusion

Cette bibliothèque me plaît bien ! Nous n'avons fait qu'effleurer ses possibilités dans ce tuto et le développement est très actif, avec une nouvelle version chaque mois quasiment. Concernant la gestion des styles, il est possible d'utiliser des fichiers yaml dans lesquels tous les paramètres sont centralisés, un peu comme des feuilles de styles CSS. Très intéressant pour réutiliser le même style pour générer plusieurs cartes.

Certains points peuvent cependant être améliorés :

- traduction des noms des constellations,
- plus de noms d'étoiles affichés
- ajout d'une image sur la carte (par exemple le logo du club astro)

D'après la documentation, pour les deux premiers points, on devrait pouvoir faire ces modifications sans changer le code de la bibliothèque. Ce sera pour un autre article sans doute.

Ludovic Grossard