

Leçon 4

Images et packages

Trouver un package

Catalogue officiel : **typst.app/universe**

Étape	Action
-------	--------

- | | |
|---|--------------------------------------|
| 1 | Aller sur typst.app/universe |
| 2 | Chercher ou parcourir les catégories |
| 3 | Lire la documentation du package |
| 4 | Copier la ligne d'import |

```
#import "@preview/nom-package:version": fonction
```

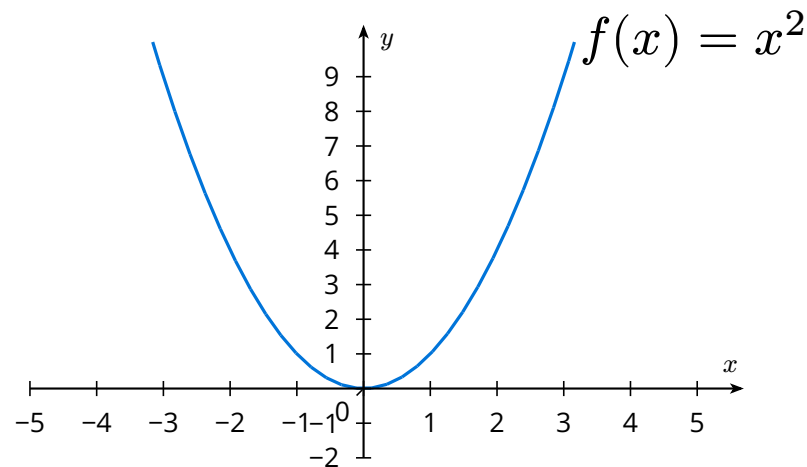
Graphiques avec simple-plot

<https://typst.app/universe/package/simple-plot/>

Code

```
#import "@preview/simple-plot:0.9.0":  
  plot  
  
#plot(  
  xmin: -5, xmax: 5.2,  
  ymin: -2, ymax: 10,  
  (fn: x => x*x, label:  $f(x) = x^2$ ,  
  label-pos: 0.8, label-side: "right"),  
)
```

Rendu



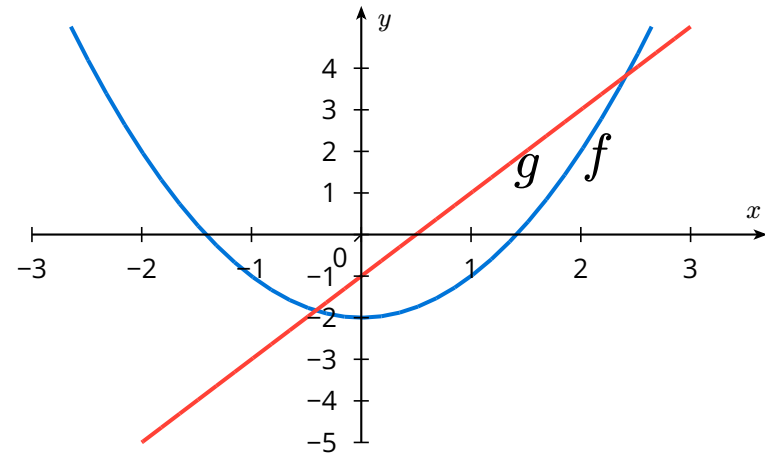
Paramètres : `xmin`, `xmax`, `ymin`, `ymax`, `width`, `height`

Plusieurs courbes

Code

```
#plot(  
  xmin: -3, xmax: 3.2,  
  ymin: -5, ymax: 5,  
  (fn: x => x*x - 2, stroke: blue +  
1.5pt, label: $f$, label-pos: 0.8,  
label-side: "right"),  
  (fn: x => 2*x - 1, stroke: red +  
1.5pt, label: $g$, label-pos: 0.7,  
label-side: "right"),  
)
```

Rendu



Chaque courbe : `fn`, `stroke` (couleur + épaisseur), `label`

Centrer les graphiques

Code

```
#align(center)[  
  #plot(...)  
]
```

Rendu

graphique centré

Attention — Toujours centrer graphiques et images avec

```
#align(center)[...]
```

Package taskize

<https://typst.app/universe/package/taskize/>

Code

```
#import "@preview/taskize:0.2.5": tasks

#tasks(columns: 2)[
  + Première question
  + Deuxième question
  + Troisième question
  + Quatrième question
]
```

Rendu

1) Première
question

3) Troisième
question

2) Deuxième
question

4) Quatrième
question

Répartition automatique en colonnes.

Options taskize

Code

```
#tasks(  
  columns: 2,  
  label-format: "a)",  
  row-gutter: 1em,  
) [  
  + Première question  
  + Deuxième question  
  + Troisième question  
  + Quatrième question  
]
```

Rendu

a) Première
question

b) Deuxième
question

c) Troisième
question

d) Quatrième
question

`columns` répartit les questions, `label-format` change les puces, `row-gutter` règle l'espacement vertical.

Package zebraw

<https://typst.app/universe/package/zebraw/>

```
#import "@preview/zebraw:0.6.3": *  
#show: zebraw  
  
\\python  
def ma_fonction(x):  
    return x * x  
\\
```

Coloration syntaxique automatique pour les blocs de code.

Insérer une image

Code

```
#image("chemin/image.png", width: 50%)

#align(center)[
  #image("photo.jpg", width: 70%)
]
```

Rendu



image.png

Paramètres : width, height, fit

Figure avec légende

Code

```
#figure(  
  image("graphique.png", width: 70%),  
  caption: [Graphique de  $f$ ],  
) <fig-graphique>
```

Voir la @fig-graphique.

Rendu



graphique

Fig. 1. – Graphique de f

Voir la Figure 1.

Tableaux simples

Code

```
#table(  
  columns: 3,  
  [A], [B], [C],  
  [1], [2], [3],  
)
```

Rendu

A	B	C
1	2	3

Tableaux avec en-tête

Code

```
#table(  
  columns: (auto, 1fr, 2fr),  
  table.header(  
    [*N°*], [*Nom*], [*Description*]  
  ),  
  [1], [Item], [Texte long],  
)
```

Rendu

N°	Nom	Description
1	Item	Texte long ici

auto : largeur automatique / 1fr, 2fr : proportions

Fusion de cellules

Code

```
#table(  
  columns: 3,  
  table.cell(colspan: 2)[Fusionné sur 2  
colonnes], [C],  
  [A], [B], [C],  
)
```

Rendu

Fusionné sur 2 colonnes		C
A	B	C

Un tableau compliqué ? Demandez à un LLM.

Grilles flexibles

Code

```
#grid(  
  columns: (1fr, 2fr),  
  gutter: 1em,  
  [Colonne 1 (1/3)],  
  [Colonne 2 (2/3)],  
)
```

Rendu

Colonne
1 (1/3)

Colonne 2 (2/3)

grid pour la mise en page ; table pour les tableaux.

Références croisées

= **Mon chapitre** <chap-intro>

#figure(...) <fig-1>

\$ E = m c^2 \$ <eq-einstein>

Voir le @chap-intro.

La @fig-1 montre le graphique.

Découverte de packages

1. Chercher `icons` sur Typst Universe
2. Ouvrir une page de package pertinente
3. Lire l'exemple minimal
4. Copier l'import depuis la documentation
5. Ajouter une icône devant un titre ou une remarque

Note — Même méthode pour l'exercice : trouver un package de **QR code** (par exemple `tiaoma`) et reprendre son exemple minimal.

Rendre un contenu cliquable

`#link(url)[contenu]` rend cliquable **tout** son contenu, y compris une image ou un QR code.

Code

```
// QR code statique  
#tiaoma.qrcode("https://typst.app")
```

Rendu

```
// QR code cliquable  
#link("https://typst.app") [  
  #tiaoma.qrcode("https://typst.app")  
]
```

Astuce — Le même `#link(...)` fonctionne autour d'un texte, d'une image ou d'un `#box`.

Checklist

- ☐ Importer les packages nécessaires
- ☐ Graphiques avec bornes adaptées
- ☐ Courbes de couleurs distinctes
- ☐ Questions en colonnes avec taskize
- ☐ Images centrées avec align
- ☐ Légendes avec figure
- ☐ QR code cliquable avec `#link`

Leçon suivante : template avancé