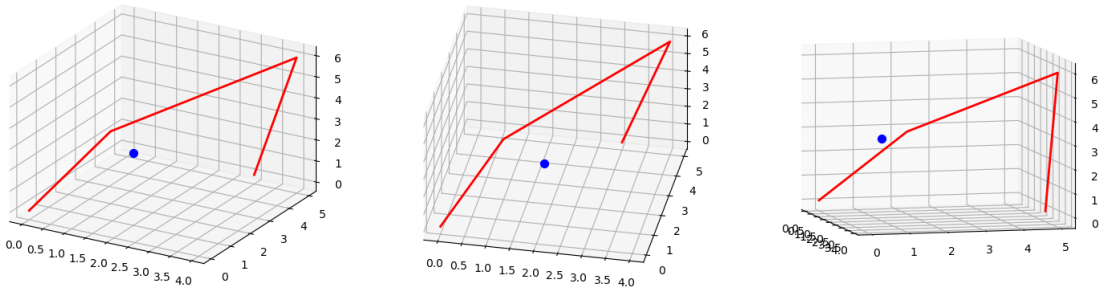
# Images 3D

*Comment dessiner des objets dans l'espace et comment les représenter sur un plan ?*

**Cours 1** (Images 3D avec matplollib)**.**
Avec le module `matplolib` il est assez facile de tracer une représentation des objets dans l'espace. Le principe est similaire à l'affichage dans le plan, sauf bien sûr qu'il faut préciser trois coordonnées $x, y, z$ pour déterminer un point de l'espace.

Voici un code très simple qui affiche :

- un point bleu de coordonnées $(2, 1, 3)$,
- des segments rouges qui relient les points de la liste $(0, 0, 0)$, $(1, 2, 3)$, $(4, 5, 6)$, $(3, 5, 0)$.



Une fenêtre s'affiche dans laquelle sont dessinés le point et les segments ainsi que les plans quadrillés de coordonnées. L'image est dynamique : à l'aide de la souris tu peux faire tourner le dessin afin de changer de point de vue.

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Initialisation
fig = plt.figure()
ax = fig.gca(projection='3d',proj_type = 'ortho')

# Affichage d'un point
x,y,z = (2,1,3)
ax.scatter(x,y,z,color='blue',s=50)

# Segments reliant des points
points  = [(0,0,0),(1,2,3),(4,5,6),(3,5,0)]
```

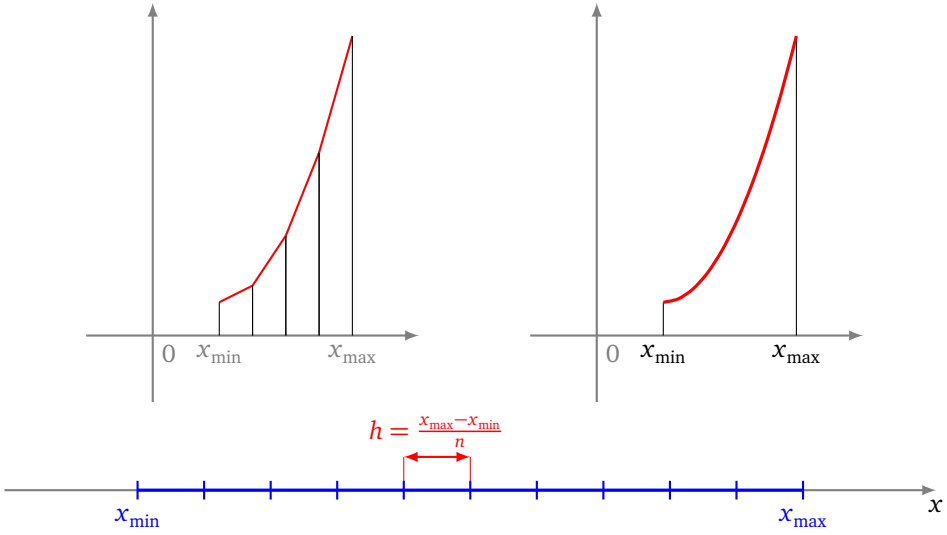
```
liste_x = [x for x,y,z in points]
liste_y = [y for x,y,z in points]
liste_z = [z for x,y,z in points]

ax.plot(liste_x,liste_y,liste_z,color='red',linewidth=2)

# Affichage
plt.show()
```

*Avertissement.* Pour afficher des segments la commande `plot` n'est pas très naturelle (mais c'était déjà le cas dans le plan). Par exemple pour relier le point $(1,2,3)$ au point $(4,5,6)$ on donne d'abord la liste des $x$, puis la liste des $y$, puis la liste des $z$ :

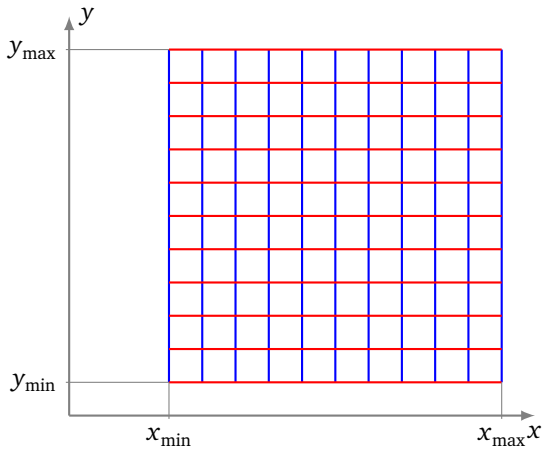
```
plot([1,4],[2,5],[3,6])
```

**Cours 2** (Surface d'équation $z = f(x,y)$)**.**

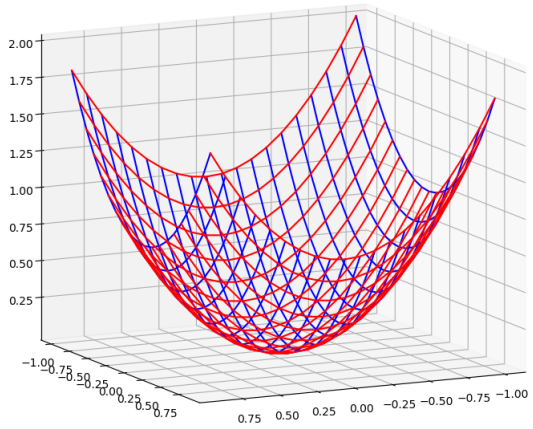
- Une ***fonction de deux variables*** associe à un couple de réels $(x,y)$ un réel $f(x,y)$, c'est donc une fonction $f : \mathbb{R}^2 \to \mathbb{R}$, $(x,y) \mapsto f(x,y)$.
- Le ***graphe*** d'une fonction de deux variables est la surface d'équation $z = f(x,y)$, autrement dit c'est
$$\{(x,y,z) \in \mathbb{R}^3 \mid z = f(x,y)\}$$
- Pour tracer le graphe d'une fonction d'une seule variable, on relie des points $(x, f(x))$ entre eux (sur la figure de gauche 5 points), si les points sont suffisamment proches, la courbe a l'air lisse (sur la figure de droite avec 20 points).




- Pour tracer une surface associée à une fonction de deux variables on commence par quadriller le plan $(x,y)$.

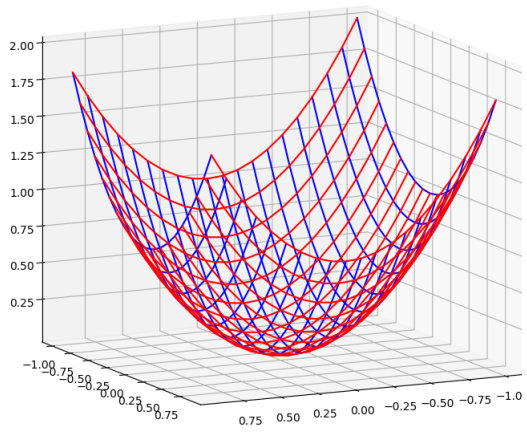



- Ensuite on trace les points $(x, y, f(x, y))$ au-dessus de chaque ligne verticale bleue du plan en les reliant puis on fait la même chose sur chaque ligne horizontale rouge.

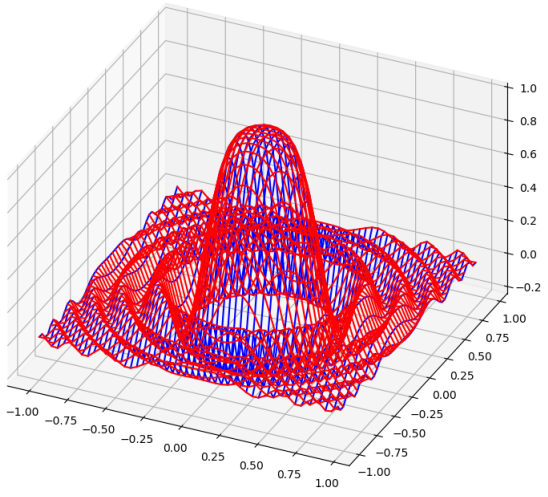


**Activité 1** (Surfaces)**.**

*Objectifs : tracer la surface d'équation $z = f(x, y)$ donnée par une fonction de deux variables.*



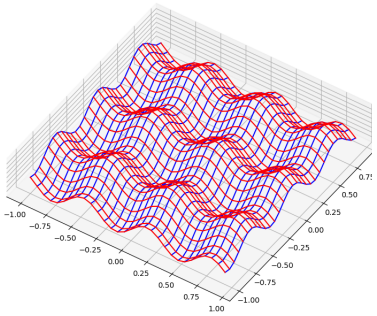
Bol – $f(x, y) = x^2 + y^2$



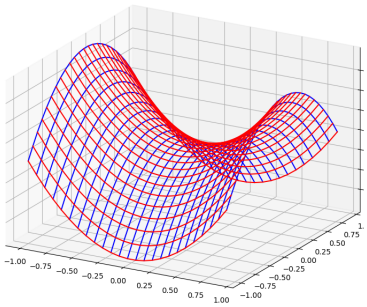
Goutte qui tombe dans l'eau – $f(x, y) = \frac{\sin(r)}{r}$ où $r = 20(x^2 + y^2)$

1. Programme une fonction `f(x,y)` qui renvoie une valeur en fonction de $x$ et de $y$. Voici des exemples de fonctions :
   - $f(x,y) = x^2 + y^2$ (un bol).
   - $f(x,y) = \frac{\sin(r)}{r}$ où $r = 20(x^2+y^2)$ (une goutte qui tombe dans l'eau).
   - $f(x,y) = \sin(10x) + \cos(10y)$ (une boîte d'œufs).
   - $f(x,y) = x^2 - y^2$ (une selle de cheval).
   - $f(x,y) = \exp(-\frac{1}{3}x^3 + x - y^2)$ pour $x \in [-2,3]$ et $y \in [-2.5,2.5]$ (un sommet et un col).

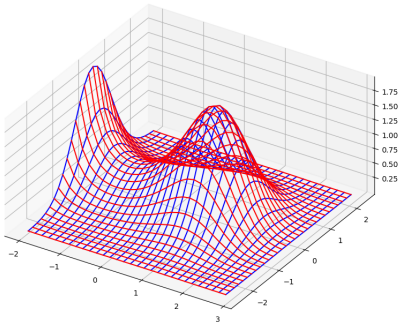
   Les graphes d'équation $z = f(x,y)$ sont dessinés ci-dessus ou ci-dessous. À l'exception de la dernière fonction, les tracés sont obtenus pour $x \in [-1,+1]$ et $y \in [-1,+1]$.



Boîte d'œufs – $f(x,y) = \sin(10x) + \cos(10y)$



Selle de cheval – $f(x,y) = x^2 - y^2$



Sommet et col – $f(x,y) = \exp(-\frac{1}{3}x^3 + x - y^2)$ pour $x \in [-2,3]$ et $y \in [-2.5,2.5]$

2. (a) Définis des constantes globales `xmin = -1`, `xmax = +1` pour définir l'intervalle $x \in [-1,+1]$ et `ymin = -1`, `ymax = +1` pour définir l'intervalle $y \in [-1,+1]$.

   Définis aussi une constante globale `nbpoints = 10` qui correspond au nombre $N$ de découpages. Chaque ligne est formée de $N+1$ points ; il y a aussi en tout $N+1$ lignes tracées dans chaque direction.

   (b) Programme une fonction `liste_points_xcst(x)` qui renvoie une liste de points $(x,y,z)$ où :
   - $x$ est la valeur donnée en paramètre de la fonction,
   - $y$ prend les valeurs $y_{\min} + kh$ pour $k = 0,1,\ldots,N$ et $h = \frac{y_{\max}-y_{\min}}{N}$ (on rappelle que $N =$ `nbpoints`),
   - $z = f(x,y)$.

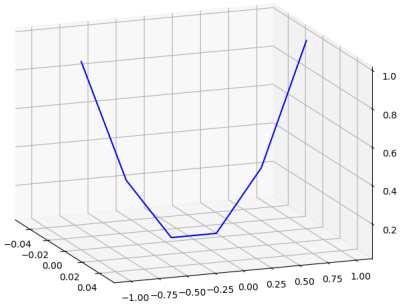
   Par exemple pour la fonction $f(x,y) = x^2 + y^2$, $x = 0$ et l'intervalle $[-1,+1]$ des $y$ découpé en $N = 5$ morceaux, la liste des points renvoyée est une liste de $N+1 = 6$ points :

```
[(0, -1, 1), (0, -0.6, 0.36), (0, -0.2, 0.04),
 (0, 0.2, 0.04), (0, 0.6, 0.36), (0, 1, 1)]
```

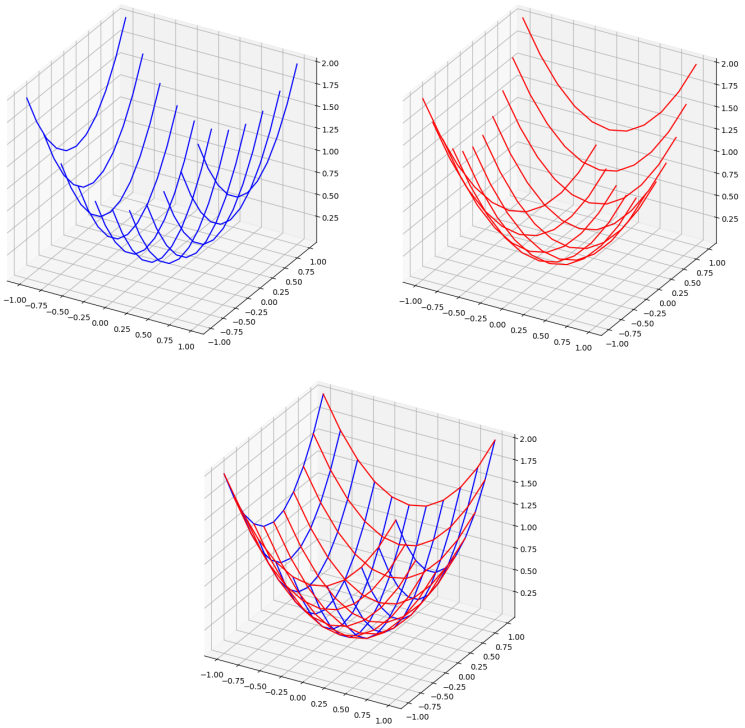
   (c) Fais le même travail pour une fonction `liste_points_ycst(y)` où cette fois $y$ est fixé et c'est $x$ qui varie.

3. (a) Programme une fonction `trace_ligne(liste_points)` (ou mieux `trace_ligne(liste_points,couleur='gray')` permettant de changer la couleur du trait) qui relie les points $(x,y,z)$ de la liste par des segments (du premier au dernier).

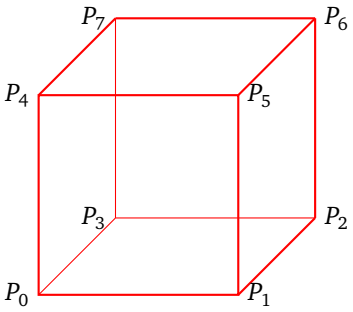
   Voici l'affichage des 6 points de la liste de la question précédente.



(b) Programme ensuite une fonction `trace_surface()` qui dessine la surface en traçant $N+1$ lignes (bleues) pour chacune desquelles $x$ est constant (figure de gauche) et aussi $N+1$ lignes (rouges) pour chacune desquelles $y$ est constant (figure du milieu) pour obtenir une représentation de la surface (figure de droite).



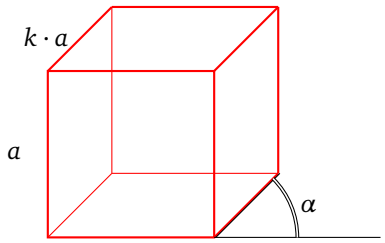
**Cours 3** (Perspective)**.**
Ceci n'est pas un cube !




C'est juste une union de segments du plan. Notre cerveau est capable de reconstruire un objet en trois dimensions à partir d'un image plane. Pour dessiner sur une feuille un objet 3D il faut donc une formule

qui transforme un point $(x, y, z)$ de l'espace en un point $(X, Y)$ du plan. Il existe différentes formules, en voici quelques unes. Pour les dessins voir l'activité qui suit.

**Perspective cavalière.**
Elle est définie par une constante $k$ qui réduit les longueurs des segments obliques et un angle $\alpha$.




$$\begin{cases} X &= x + k\cos(\alpha)y \\ Y &= z + k\sin(\alpha)y \end{cases}$$

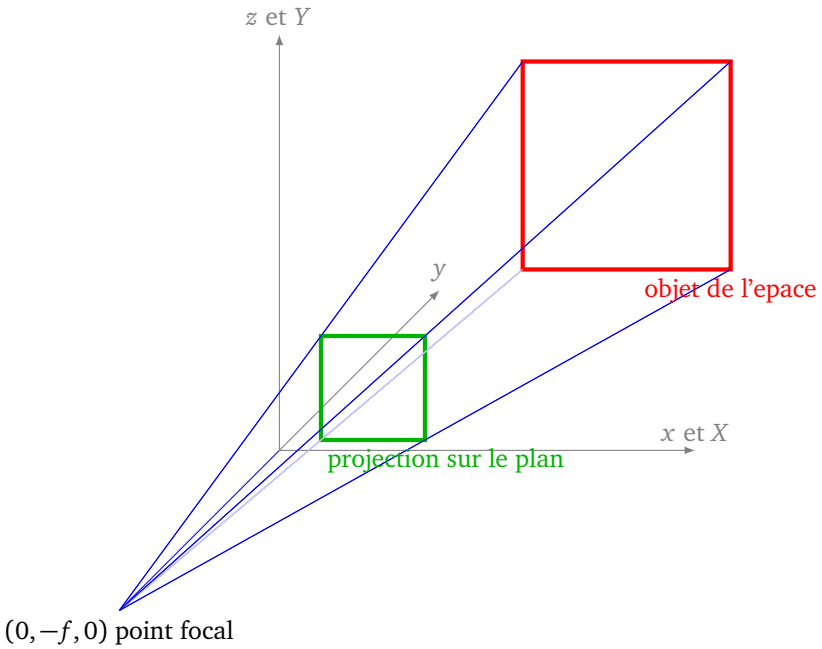
Les constantes $\alpha$ et $k$ sont le plus souvent $(\alpha = \frac{\pi}{4}, k = \frac{1}{2})$ ou bien $(\alpha = \frac{\pi}{6}, k = 0.7)$.

**Perspective axonométrique.**
Cette opération consiste à tourner d'abord l'objet selon deux axes (il y a donc deux angles $\omega$ et $\alpha$), avant de projeter sur un plan.

$$\begin{cases} X &= \cos(\omega)x - \sin(\omega)y \\ Y &= -\sin(\omega)\sin(\alpha)x - \cos(\omega)\sin(\alpha)y + \cos(\alpha)z \end{cases}$$

Où $\omega$ et $\alpha$ sont des angles fixés. Dans le cas particulier $\omega = 0.61$ et $\alpha = \frac{\pi}{4}$ on obtient la *perspective isométrique*.

**Perspective conique.**
Il s'agit de regarder un objet de l'espace depuis un point $(0, -f, 0)$ et de le projeter sur le plan $(y = 0)$ ; $f$ est une valeur constante qui s'appelle la *focale*.




Les formules sont :

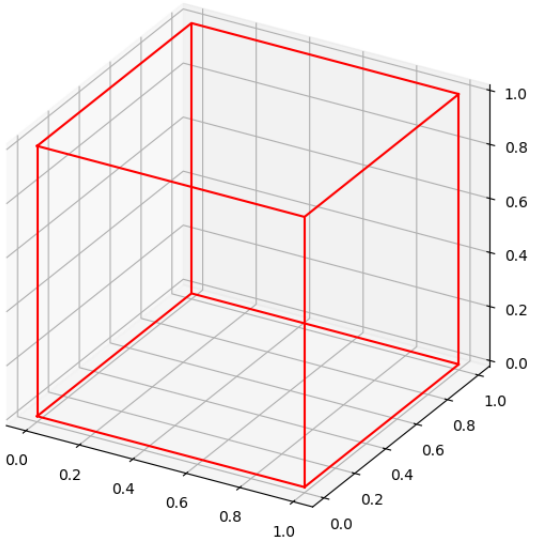
$$\begin{cases} X &= kx \\ Y &= kz \end{cases} \qquad \text{avec} \quad k = \frac{f}{y+f}.$$

**Activité 2** (Perspective)**.**

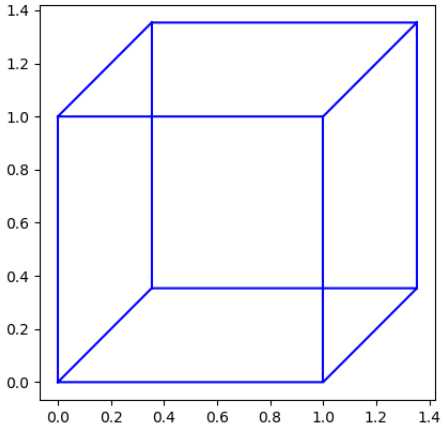
*Objectifs : transformer un objet de l'espace en un objet du plan afin de l'afficher.*

Pour les exemples on va afficher différentes projections du cube donné par les coordonnées suivantes :
`cube = [(0,0,0),(1,0,0),(1,1,0),(0,1,0),(0,0,1),(1,0,1),(1,1,1),(0,1,1)]`

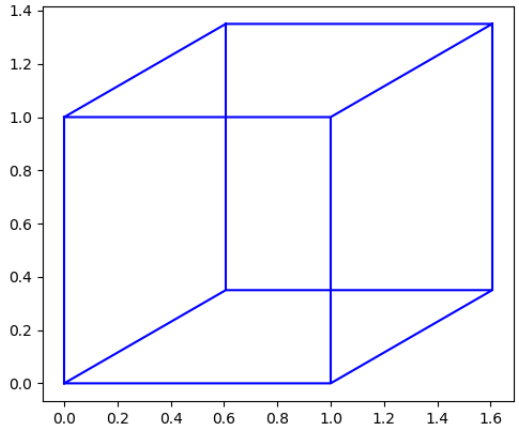
1. Programme une fonction `affiche_cube_3d(cube)` qui à partir d'une liste de 8 points $[P_0, P_1, \dots, P_7]$ trace l'affichage 3D du cube $(P_0, P_1, \dots, P_7)$. Il s'agit juste de tracer les 12 arêtes du cube !




2. **Perspective cavalière.**
   (a) Programme une fonction `perspective_cavaliere(P)` (ou mieux `perspective_cavaliere(P,alpha=pi/4,k=0.5))` qui à partir d'un point $P$ de l'espace de coordonnées $(x, y, z)$ renvoie le point $Q$ du plan de coordonnées $(X, Y)$ selon la formule :
   $$\begin{cases} X &= x + k\cos(\alpha)y \\ Y &= z + k\sin(\alpha)y \end{cases}$$
   Pour les constantes $\alpha$ et $k$, on testera $(\alpha = \frac{\pi}{4}, k = \frac{1}{2})$ puis $(\alpha = \frac{\pi}{6}, k = 0.7)$.
   (b) Programme une fonction `affiche_cube_2d(cube2d)` qui à partir d'une liste $[Q_0, Q_1, \dots, Q_7]$ de 8 **points du plan**, relie les points deux à deux comme si c'était les arêtes d'un cube.
   (c) À partir des sommets du cube 3D $(P_0, P_1, \dots, P_7)$, calcule sa projection $(Q_0, Q_1, \dots, Q_7)$ dans le plan et affiche cette projection du cube. Voici le résultat ci-dessous : il s'agit bien ici de deux images du plan !



$\alpha = \frac{\pi}{4}, k = \frac{1}{2}$

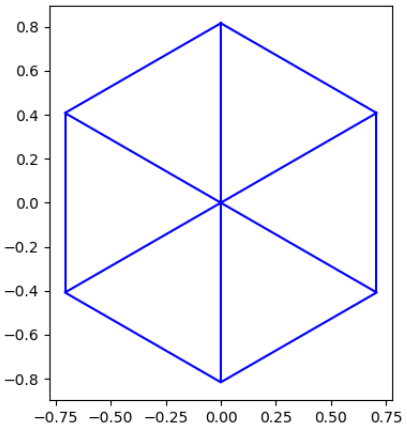


$\alpha = \frac{\pi}{6}, k = 0.7$

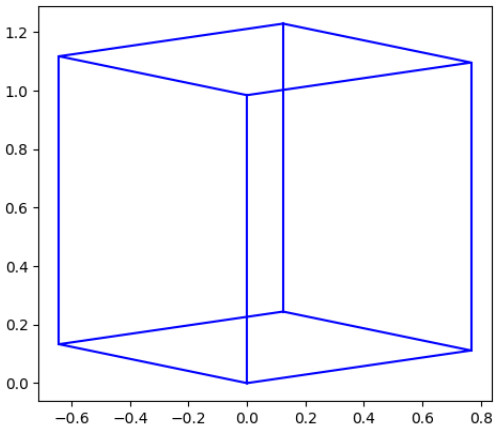
3. **Perspective axonométrique.** Fais le même travail pour une fonction `perspective_axonometrique(P)` (ou mieux `perspective_axonometrique(P, alpha=0.61, omega=pi/4)`) qui pour $P = (x, y, z)$ point de l'espace renvoie le point $Q = (X, Y)$ du plan suivant les formules :

$$\begin{cases} X &= \cos(\omega)x - \sin(\omega)y \\ Y &= -\sin(\omega)\sin(\alpha)x - \cos(\omega)\sin(\alpha)y + \cos(\alpha)z \end{cases}$$

Affiche la projection du cube.



Perspective isométrique : $\omega = 0.61$, $\alpha = \frac{\pi}{4}$



Perspective axonométrique avec $\omega = 30°$, $\alpha = -10°$ (à convertir en radians)

Avec la perspective isométrique (à gauche) toutes les arêtes projetées ont la même longueur. Ici la projection n'est pas très lisible car deux sommets sont projetés sur le même point.

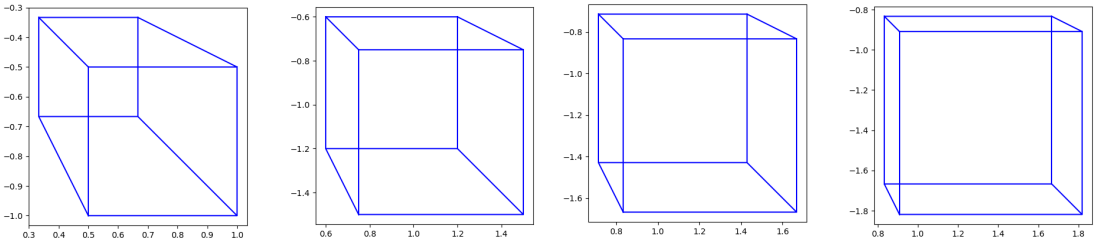
4. **Perspective conique.** Fais le même travail pour une fonction `perspective_conique(P)` (ou mieux `perspective_conique(P,f=2)`) qui pour $P = (x, y, z)$ point de l'espace renvoie le point $Q = (X, Y)$ du plan suivant les formules :

$$\begin{cases} X &= kx \\ Y &= kz \end{cases} \qquad \text{avec} \quad k = \frac{f}{y + f}.$$

Voici l'affichage du cube :

```
cube = [(1,1,-1), (2,1,-1), (2,2,-1), (1,2,-1), (1,1,-2), (2,1,-2),
                    (2,2,-2), (1,2,-2)]
```

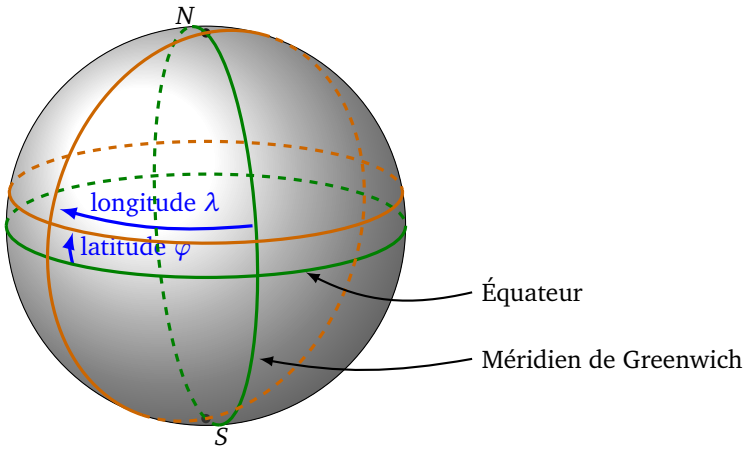
pour différentes valeurs de la focale $f$.



$f = 1$ $f = 3$ $f = 5$ $f = 10$

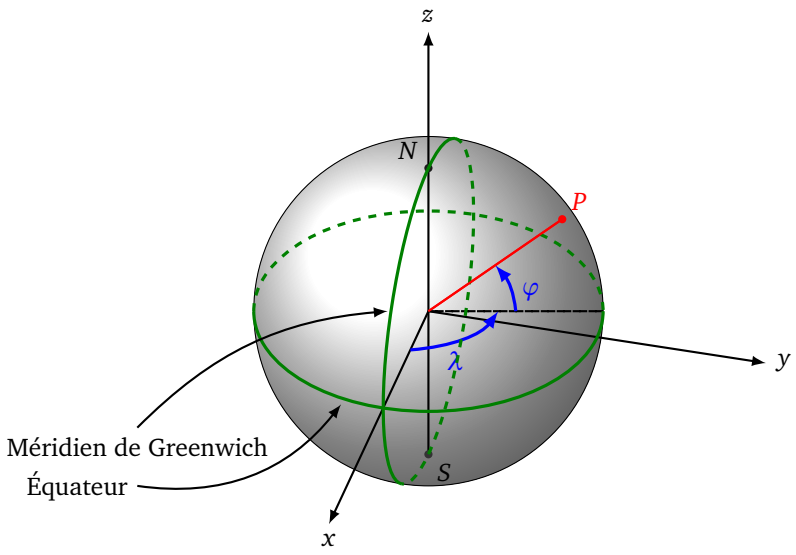
**Cours 4** (Coordonnées sur la sphère : latitude et longitude)**.**
Pour se repérer dans l'espace on peut utiliser les coordonnées $(x, y, z)$ d'un repère orthonormé direct. Mais pour se repérer à la surface de la Terre ou plus généralement sur une sphère, on peut aussi utiliser les ***coordonnées sphériques*** $[r : \varphi : \lambda]$ où :

- $r > 0$ est le rayon de la sphère,
- $\varphi$ est la ***latitude***, c'est un angle de $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ (autrement dit compris entre $-90°$ et $+90°$),
- $\lambda$ est la ***longitude***, c'est un angle de $]-\pi, +\pi]$ (autrement dit compris entre $-180°$ et $+180°$).




**Passage vers les coordonnées cartésiennes.**
On trouve $(x, y, z)$ en fonction de $[r : \varphi : \lambda]$ par les formules suivantes :

$$\begin{cases} x &= r\cos\varphi\cos\lambda \\ y &= r\cos\varphi\sin\lambda \\ z &= r\sin\varphi \end{cases}$$




**Passage vers les coordonnées sphériques.**
Pour trouver $[r : \varphi : \lambda]$ à partir de $(x, y, z)$ c'est un peu plus compliqué.
Le rayon

$$r = \sqrt{x^2 + y^2 + z^2}$$

La latitude

$$\varphi = \arcsin\left(\frac{z}{r}\right)$$

La longitude

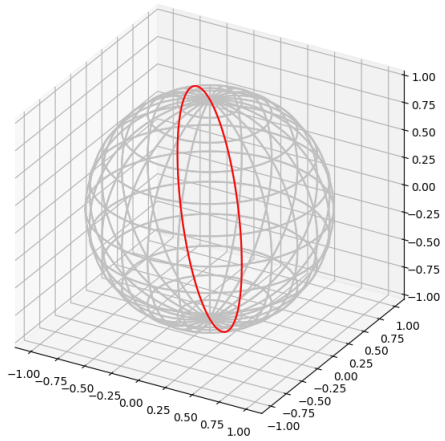
$$\lambda = \arcsin\left(\frac{1}{\cos\varphi}\frac{y}{r}\right)$$

**Activité 3** (Coordonnées sur la sphère : latitude et longitude)**.**

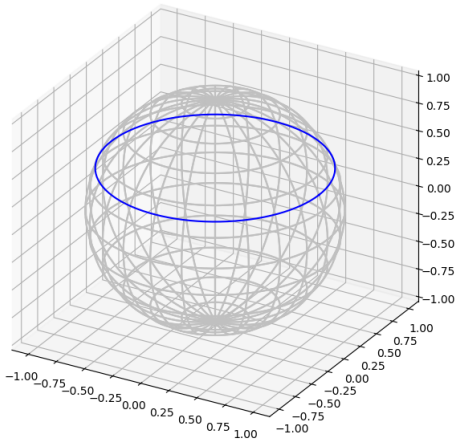
*Objectifs : se repérer sur la sphère grâce à la latitude et la longitude.*

1. **Vers les coordonnées cartésiennes.**
   Programme une fonction `latlong_vers_xyz(r,phi,lamb)` qui renvoie les coordonnées cartésiennes $(x, y, z)$ du point de coordonnées sphériques $[r : \varphi : \lambda]$.
   *Attention.* Ne pas utiliser une variable nommée `lambda` qui est un nom réservé par `Python` pour autre chose.
   *Question.* Quelles sont les coordonnées $(x, y, z)$ du point vérifiant $r = 1$, de latitude $\varphi = 45°$ et de longitude $\lambda = 30°$. (N'oublie pas de convertir les degrés en radians.)
2. **Vers les coordonnées sphériques.**
   Programme une fonction `xyz_vers_latlong(x,y,z)` qui renvoie les coordonnées sphériques $[r : \varphi : \lambda]$ connaissant ses coordonnées cartésiennes $(x, y, z)$.
   *Question.* Quelles sont les coordonnées sphériques $[r : \varphi : \lambda]$ du point $(x, y, z) = (1, 2, 3)$ ? Vérifie que si, à partir de ces $[r : \phi : \lambda]$ et de la première question, tu calcules $(x, y, z)$ tu retrouves bien $(1, 2, 3)$.
3. **Tracer les méridiens et les parallèles.**
   (a) Programme une fonction `trace_meridien(r,lamb)` (ou mieux `trace_meridien(r,lamb,nbpoints=100,couleur='red')`) qui trace le cercle méridien, connaissant le rayon $r$ et la longitude $\lambda$. (Figure de gauche ci-dessous.)
   *Indications.*

- Définis $N$ points $[r : \varphi : \lambda]$ où $\varphi$ varie dans $[-\pi, \pi]$.
- Calcule les coordonnées $(x, y, z)$ de chacun de ces points.
- Relie les points entre eux.



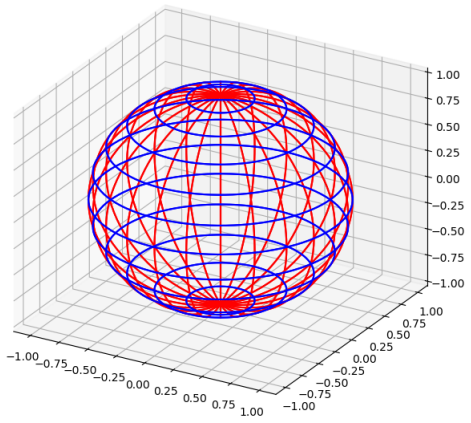
*Un méridien.*



*Un parallèle.*

(b) Programme une fonction `trace_parallele(r,phi)` (ou mieux `trace_parallele(r,phi,nbpoints=100,couleur='blue')`) qui trace le cercle parallèle, connaissant le rayon $r$ et la latitude $\varphi$. (Figure de droite ci-dessus.)

(c) Programme une fonction `trace_meridiens_parallelles(r)` qui trace des méridiens et des parallèles sur la sphère de rayon $r$.



4. **Grand cercle passant par deux points.**

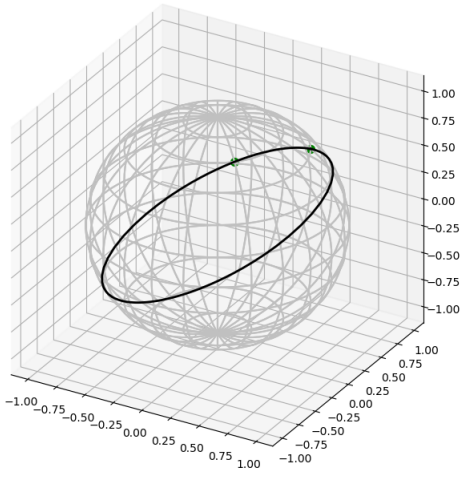
*Problème.* Quel est le trajet que doit parcourir un avion entre deux villes de la Terre ?

Mathématiquement, on se donne deux points $P$ et $Q$ sur la sphère de rayon $r$. On cherche le chemin le plus court tracé à la surface de la sphère qui va de $P$ à $Q$. Réponse : c'est un des arcs du « grand cercle » passant par $P$ et $Q$. Un *grand cercle* est un cercle de rayon $r$ tracé sur la sphère ayant ce même rayon $r$ (l'équateur et les méridiens sont des exemples de grands cercles).

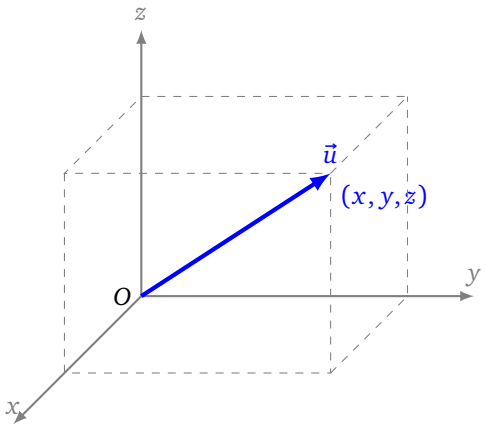
Voici comment tracer ce grand cercle :

- On considère les points $P$ et $Q$ comme des vecteurs : $\vec{u} = \overrightarrow{OP}$ et $\vec{v} = \overrightarrow{OQ}$.
- On considère le vecteur tournant $\vec{w}(t) = \cos(t)\vec{u} + \sin(t)\vec{v}$, pour $t \in [0, 2\pi]$.
- On transforme le vecteur $\vec{w}(t)$ en un vecteur de norme $r$ : $\vec{w}'(t) = r\frac{\vec{w}(t)}{\|\vec{w}(t)\|}$.
- Le point $R(t)$ à l'extrémité de $\vec{w}'(t)$ est sur le grand cercle passant par $P$ et $Q$ (autrement dit $R(t)$ est tel que $\vec{w}'(t) = \overrightarrow{OR(t)}$).

Programme une fonction `trace_grand_cercle(P,Q)` qui affiche le grand cercle passant par les deux points $P$ et $Q$.

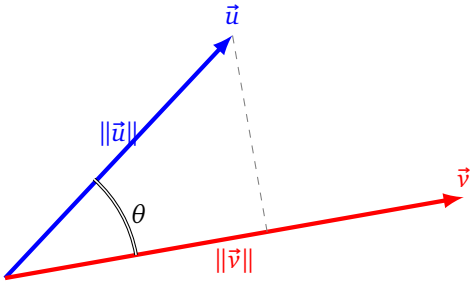



**Cours 5** (Vecteurs)**.**
On se place dans un repère orthonormé direct $(O, \vec{i}, \vec{j}, \vec{k})$. Un vecteur $\vec{u}$ est représenté par trois coordonnées $(x, y, z)$.



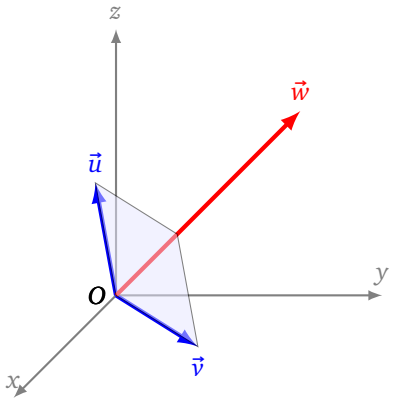

Soient $\vec{u} = (x, y, z)$, $\vec{v} = (x', y', z')$ deux vecteurs.

- **Addition.** Le vecteur $\vec{u} + \vec{v}$ a pour coordonnées $(x + x', y + y', z + z')$.
- **Multiplication par un scalaire.** Soit $k \in \mathbb{R}$. Alors $k\vec{u}$ a pour coordonnées $(kx, ky, kz)$.
- **Produit scalaire.** $\vec{u} \cdot \vec{v} = xx' + yy' + zz'$. C'est un nombre réel qui mesure la colinéarité des vecteurs $\vec{u}$ et $\vec{v}$.




- **Norme.** $\|\vec{u}\| = \sqrt{x^2 + y^2 + z^2}$. On a aussi $\|\vec{u}\| = \sqrt{\vec{u} \cdot \vec{u}}$. La norme est la longueur du vecteur $\vec{u}$.
- **Vecteurs orthogonaux.** Deux vecteurs $\vec{u}$ et $\vec{v}$ sont orthogonaux si et seulement si $\vec{u} \cdot \vec{v} = 0$.

- **Angle entre deux vecteurs.** La formule suivante permet de calculer l'angle $\theta$ entre deux vecteurs $\vec{u}$ et $\vec{v}$ non nuls :
$$\vec{u} \cdot \vec{v} = \|\vec{u}\| \, \|\vec{v}\| \cos(\theta)$$
Ainsi
$$\theta = \arccos\left(\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|}\right)$$
- **Produit vectoriel.** $\vec{u} \wedge \vec{v}$ est le vecteur de coordonnées
$$\left(yz' - y'z, zx' - z'x, xy' - x'y\right)$$
Ce vecteur est orthogonal au vecteur $\vec{u}$ et au vecteur $\vec{v}$. Autrement dit $\vec{w} = \vec{u} \wedge \vec{v}$ est orthogonal au plan qui contient $\vec{u}$ et $\vec{v}$.

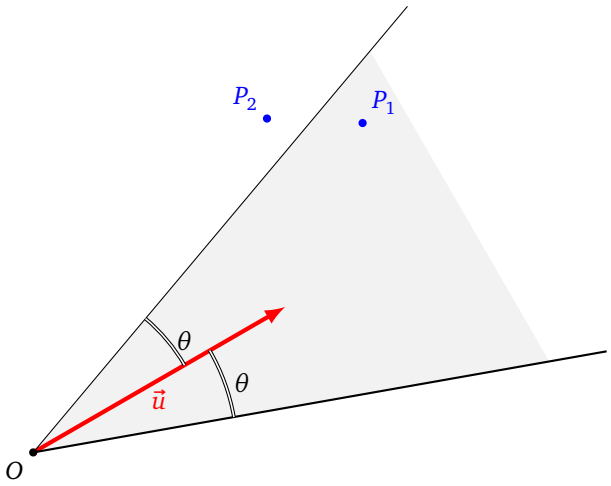



- **Produit mixte.** C'est le nombre réel associé à trois vecteurs $\vec{u}$, $\vec{v}$ et $\vec{w}$, défini à l'aide d'un produit vectoriel puis d'un produit scalaire :
$$(\vec{u} \wedge \vec{v}) \cdot \vec{w}$$

**Activité 4** (Vecteurs).

*Objectifs : utiliser les vecteurs pour des calculs dans l'espace.*

1. **Norme et produit scalaire.**
   (a) Programme une fonction `produit_scalaire(u,v)` qui calcule le produit scalaire de deux vecteurs $\vec{u}$ et $\vec{v}$ de $\mathbb{R}^3$. La variable `u` contient un triplet `(x,y,z)` représentant les coordonnées de $\vec{u}$, de même pour `v`.
   (b) Programme une fonction `norme(u)` qui calcule la norme d'un vecteur $\vec{u}$ de $\mathbb{R}^3$.
   (c) Programme une fonction `angle(u,v)` qui calcule l'angle entre des vecteurs $\vec{u}$ et $\vec{v}$.
   (d) Calcule la norme de $\vec{u} = (1,2,3)$ et de $\vec{v} = (1,0,1)$. Puis calcule le produit scalaire entre ces deux vecteurs, et enfin l'angle entre ces vecteurs (en radians et en degrés).
   (e) *Application : points visibles ou invisibles.* Un observateur regarde dans une direction, selon son champ de vision, quels sont les points visibles ?
   - *Modélisation.* L'observateur est au point $O = (0,0,0)$. Il regarde dans la direction $\vec{u}$. Son champ de vision est déterminé par un angle $\theta$. Sur le schéma ci-dessous le point $P_1$ est visible par l'observateur mais pas le point $P_2$. Bien sûr, dans l'espace, la zone visible est un cône (et pas un secteur comme sur le schéma).




- *Solution.* Un point $P$ est visible depuis $O$ si et seulement si l'angle entre $\vec{u}$ et $\overrightarrow{OP}$ est plus petit que $\theta$, c'est-à-dire :

$$\left|\text{angle}(\vec{u}, \overrightarrow{OP})\right| \leqslant \theta$$

- *Question.* Un observateur a un angle de vision $\theta = 50°$, il regarde dans la direction $(1,1,1)$. Parmi les points $P_1 = (1,1,2)$, $P_2 = (-1,-1,-2)$, $P_3 = (80,10,0)$, $P_4 = (85,10,0)$ lesquels sont visibles à ses yeux ?

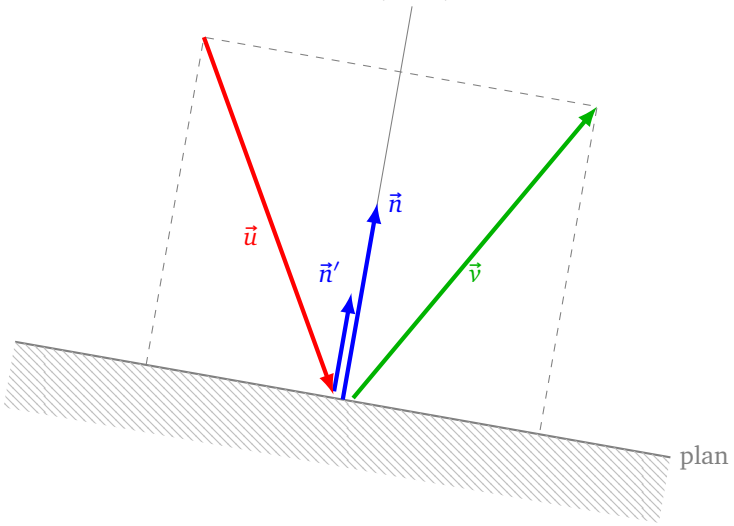
(f) *Application : rebond sur un plan.* Une balle arrive sur un plan et rebondit. Quelle est la nouvelle trajectoire de cette balle ?

- *Modélisation.* La balle arrive selon un vecteur vitesse $\vec{u}$, le plan est représenté par un vecteur normal $\vec{n}$.
- *Solution.* La balle repart selon le vecteur $\vec{v}$ qui est un symétrique de $\vec{u}$ par rapport à $\vec{n}$ et est donné par la formule suivante : on commence par transformer $\vec{n}$ en un vecteur de norme 1 :

$$\vec{n}' = \frac{\vec{n}}{\|\vec{n}\|}$$

puis on a :

$$\vec{n} = \vec{u} - 2(\vec{u} \cdot \vec{n}')\vec{n}'$$




- *Question.* Une balle arrive selon le vecteur $\vec{u} = (1,2,-1)$ et rebondit sur un plan ayant pour vecteur normal $\vec{n} = (1,1,1)$. Selon quelle direction $\vec{v}$ repart-elle ?

2. **Produit vectoriel.**

(a) Programme une fonction `produit_vectoriel(u,v)` qui calcule le produit vectoriel entre deux vecteurs $\vec{u}$ et $\vec{v}$ de $\mathbb{R}^3$. Le résultat est un vecteur $\vec{w}$ renvoyé sous la forme d'un triplet de coordonnées $(x, y, z)$.

(b) Calcule le produit vectoriel $\vec{w}$ de $\vec{u} = (1, 2, 3)$ et $\vec{v} = (1, 0, 1)$. Vérifie à l'aide du produit scalaire que $\vec{w}$ est orthogonal à $\vec{u}$ et à $\vec{v}$.

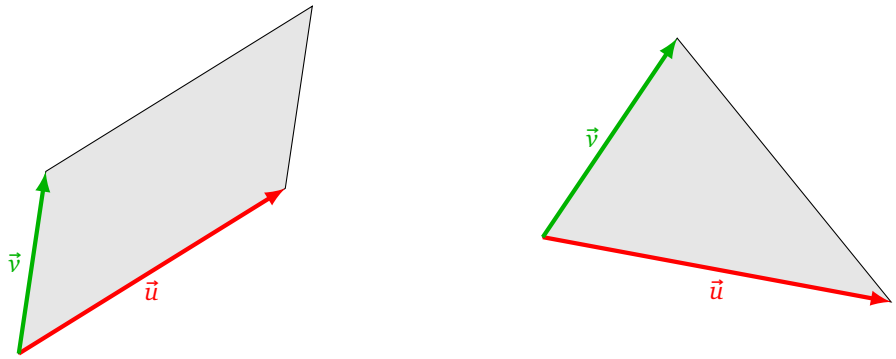
(c) *Application : équation d'un plan.* On considère le plan $P$ défini par les trois points $O = (0, 0, 0)$, $A = (-1, 2, 5)$, $B = (2, 0, 3)$. Calcule un vecteur $\vec{n} = (a, b, c)$ normal à ce plan $P$. Une équation du plan est alors $ax + by + cz = 0$.

(d) *Application : surface d'un triangle et d'un parallélogramme de l'espace.* La surface d'un parallélogramme de l'espace déterminé par deux vecteurs $\vec{u}$ et $\vec{v}$ est

$$S_P = \|\vec{u} \wedge \vec{v}\|$$

La surface du triangle de l'espace déterminé par ces mêmes vecteurs est la moitié :

$$S_T = \frac{1}{2}\|\vec{u} \wedge \vec{v}\|$$




Calcule la surface du parallélogramme (puis du triangle) déterminé par les vecteurs $\vec{u} = (1, 2, -5)$ avec $\vec{v} = (1, -2, 4)$ (il est préférable de donner la réponse à l'aide de la racine carrée d'un entier plutôt qu'une valeur approchée).

3. **Produit mixte.**

(a) Programme une fonction `produit_mixte(u,v,w)` qui calcule le produit mixte de trois vecteurs $\vec{u}$, $\vec{v}$, $\vec{w}$ de $\mathbb{R}^3$. On rappelle la formule du produit mixte

$$(\vec{u} \wedge \vec{v}) \cdot \vec{w}$$

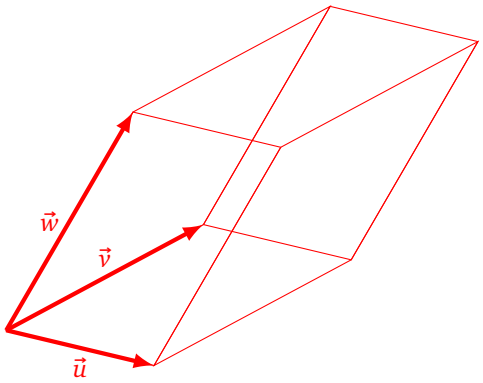
(b) Calcule le produit mixte de $\vec{u} = (1, 2, 3)$, $\vec{v} = (1, 0, 1)$ et $\vec{w} = (4, 1, 0)$.

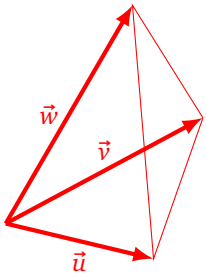
(c) *Application : volume.* Le volume d'un parallélépipède de l'espace déterminé par trois vecteur $\vec{u}$, $\vec{v}$ et $\vec{w}$ est

$$V_P = \left|\text{produit\_mixte}(\vec{u}, \vec{v}, \vec{w})\right|$$

Le volume du tétraèdre déterminé par ces mêmes vecteurs est $1/6$ du volume précédent :

$$V_T = \frac{1}{6}\left|\text{produit\_mixte}(\vec{u}, \vec{v}, \vec{w})\right|$$



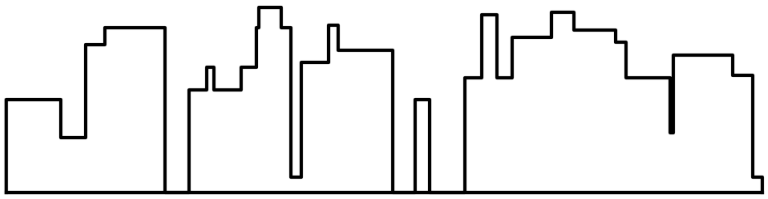




Calcule le volume du parallélépipède (puis du tétraèdre) déterminé par les vecteurs $\vec{u} = (1,0,0)$, $\vec{v} = (1,1,0)$ et $\vec{w} = (1,1,1)$.

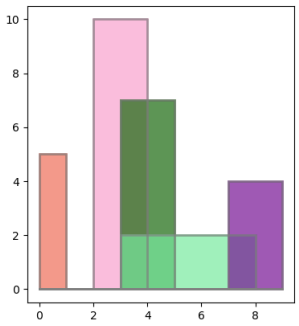
**Activité 5** (Skyline)**.**

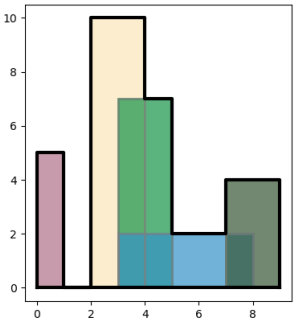
*Objectifs : tracer la* skyline *d'une ville, c'est-à-dire le contour apparent de ses gratte-ciels.*

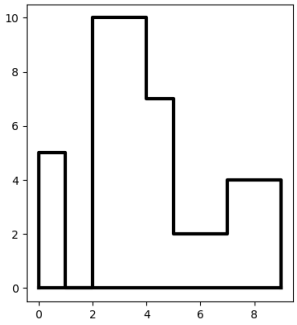
Le problème est simple : on souhaite dessiner le contour apparent d'une ville constituée de gratte-ciels.



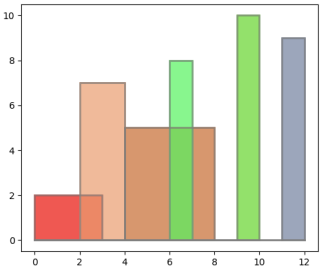
Voici à gauche les immeubles, au centre on dessine le contour, à droite on garde uniquement ce contour.

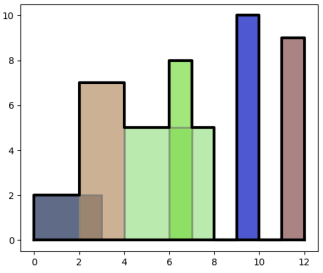


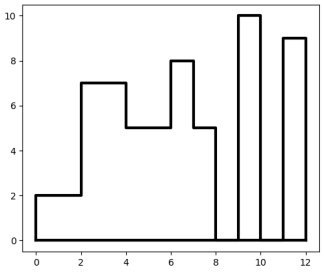




Voici un autre exemple.







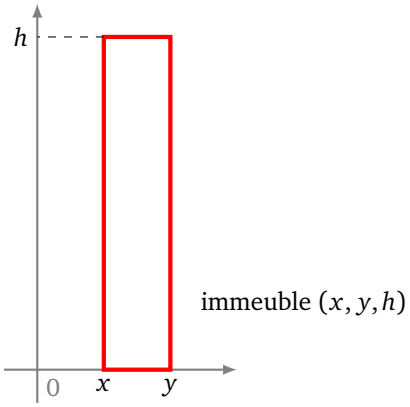
Ce qui serait formidable c'est de ne pas lire la suite de l'activité et que tu te débrouilles tout seul pour modéliser et résoudre ce problème !

*Indications.*

- Il faut d'abord comprendre que ce problème, qui devrait être un problème en trois dimensions, se ramène à un problème à deux dimensions seulement : qu'un immeuble soit devant ou derrière un autre ne change pas le contour.
- Il faut aussi avoir conscience que même si le problème est facile à énoncer, trouver une solution n'est pas simple !
- C'est plus important que tu trouves un algorithme seul, même si il n'est pas parfait, plutôt que de suivre la solution proposée ci-dessous qui est assez optimale.

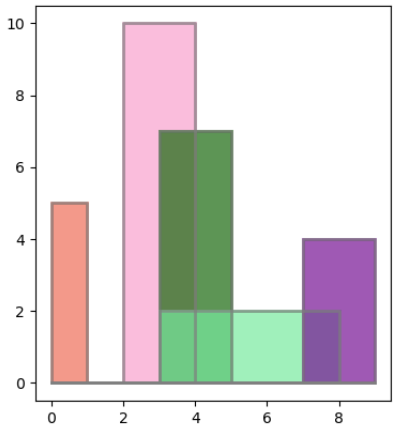
**Modélisation.**

- On modélise un gratte-ciel par un triplet $(x, y, h)$ où $x$ est l'abscisse du côté gauche de l'immeuble, $y$ est l'abscisse du côté droit, $h$ est sa hauteur.




- Une ville est donc une liste d'immeubles. Voici l'exemple pour cette activité :

```
immeubles = [(0,1,5), (2,4,10), (3,5,7), (3,8,2), (7,9,4)]
```




**Principe.** L'idée mise en œuvre ci-dessous est la suivante.

- Tout d'abord on récupère tous les bords : ce sont les $x$ ou $y$ correspondant à un côté d'immeuble.
- Pour chacun de ces bords, on associe la liste des immeubles qui commencent ou finissent ici. On utilisera un dictionnaire.
- Pour calculer la *skyline*, on parcourt les bords de la gauche vers la droite. On calcule les listes de tous les immeubles actifs : ce sont les immeubles présents à cette abscisse.
- Chaque bord détermine donc la hauteur maximale actuelle. Si cette hauteur diffère de celle du bord précédent, on a un point de la *skyline* !

Voici le travail décomposé en étapes.

1. Programme une fonction `hauteur_max_immeubles(immeubles)` qui renvoie la hauteur maximale d'une liste d'immeubles (ou 0 si la liste est vide).

   Par exemple avec :

   ```
   immeubles = [(0,1,5), (2,4,10), (3,5,7), (3,8,2), (7,9,4)]
   ```

   la fonction renvoie la hauteur maximale $h = 10$.

2. Programme une fonction `calcul_bords(immeubles)` qui renvoie la liste de tous les bords : dans l'ordre croissant et sans redondance. Un *bord* est une abscisse $x$ ou $y$ d'un immeuble.

   Par exemple avec :

   ```
   immeubles = [(0,1,5), (2,4,10), (3,5,7), (3,8,2), (7,9,4)]
   ```

   la fonction renvoie la liste :

   ```
   [0, 1, 2, 3, 4, 5, 7, 8, 9]
   ```

3. Programme une fonction `dictionnaire_bords_immeubles(immeubles)` qui renvoie un dictionnaire associant à chaque abscisse la liste des numéros d'immeubles ayant un bord ici. Une clé est donc un bord, la valeur une liste de numéros d'immeubles.

   Par exemple avec :

   ```
   immeubles = [(0,1,5), (2,4,10), (3,5,7), (3,8,2), (7,9,4)]
   ```

   la fonction renvoie le dictionnaire :

   dico = {`0: [0], 1: [0], 2: [1], 4: [1],`
   3 : [2, 3], 5 : [2], 8 : [3], 7 : [4], 9 : [4]}

   Par exemple `dico[2]` vaut `[1]` cela signifie que seul l'immeuble numéro 1 a un bord à l'abscisse 2. Autre exemple `dico[3]` vaut `[2, 3]`, cela veut dire que les immeubles numéros 2 et 3 ont un bord à l'abscisse 3.

   *Indications.*
   - La manipulation d'un dictionnaire est expliquée dans la fiche « Le mot le plus long ».
   - Pars d'un dictionnaire vide `dico = {}`.
   - Pour chaque immeuble récupère les valeurs $x$ et $y$ :
     — si $x$ n'est pas déjà une clé (test « `x not in dico` ») crée une nouvelle liste contenant le numéro de l'immeuble (`dico[x] = [i]`),
     — si $x$ est déjà une clé (test « `x in dico` ») ajoute le numéro de l'immeuble à la liste (`dico[x].append(i)`),
     — fais la même chose avec $y$.

4. Programme une fonction `calcul_skyline(immeubles)` qui renvoie la liste des points formant le contour. Pour l'exemple des questions précédentes le contour est :

   ```
   [(0, 0), (0, 5), (1, 5), (1, 0), (2, 0), (2, 10), (4, 10),
    (4, 7), (5, 7), (5, 2), (7, 2), (7, 4), (9, 4), (9, 0)]
   ```

   Voici l'algorithme :
   - Calculer la liste des bords et le dictionnaire bords/immeubles.
   - Initialiser une liste vide pour la *skyline* et une pour les immeubles actifs.
   - Initialiser une hauteur `h_avant` à 0.
   - Pour chaque $x$ dans la liste des bords :
     — calculer la liste des immeubles actifs (pour chaque $i$ dans `dico[x]` on ajoute ou on retire l'immeuble numéro $i$ s'il est absent ou déjà présent),
     — calculer la nouvelle hauteur maximale `h_apres` des immeubles actifs,
     — si `h_avant` et `h_apres` diffèrent alors ajouter à la *skyline* les deux points `(x,h_avant)` et `(x,h_apres)`,
     — `h_avant` ← `h_apres`

5. Programme enfin l'affichage des immeubles et de la *skyline* !

Voici un exemple généré au hasard.