TP14 En Python
Rosace parabolique : Fonction Carré

Allumer l'ordinateur et connectez-vous en utilisant votre login et votre mot de passe puis lancer **«** Python en ligne : https://www.codabrainy.com/python-compiler/ »

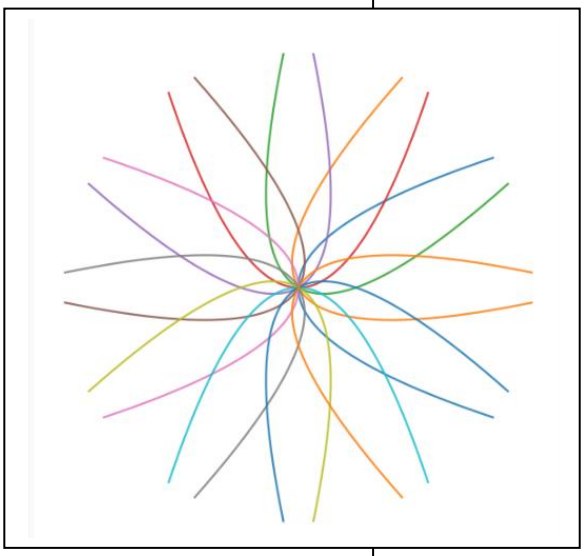
Activité

On réutilise les paraboles horizontales étudiées dans **le TP 13** comme des modules géométriques de base. L'idée est de ne plus les voir seulement comme des courbes isolées, mais comme des formes qu'on peut transformer : on applique des symétries, des rotations et des superpositions pour construire une composition artistique inspirée de l'art génératif. Chaque transformation ajoute une couche de structure visuelle, un peu comme si la géométrie devenait un langage décoratif.

Algorithme N°5 « Rosace parabolique »

Rosace parabolique : on prend une parabole horizontale et on la répète par rotations successives autour de l'origine. Chaque copie agit comme un "pétale". L'ensemble forme une rosace circulaire proche d'un mandala, où la régularité mathématique crée un motif floral équilibré et harmonieux.

```
# Algorithme N°5 (artistique) : Rosace parabolique par rotations
import numpy as np
import matplotlib.pyplot as plt

def parabole_droite(y, a=0.35, c=0.0):
    # x = a*y^2 + c  (branche vers la droite)
    return a * y**2 + c

def rotation(x, y, theta):
    # Rotation d'angle theta autour de l'origine
    ct, st = np.cos(theta), np.sin(theta)
    xr = ct * x - st * y
    yr = st * x + ct * y
    return xr, yr

# Domaine
y = np.linspace(-6, 6, 1200)
x = parabole_droite(y, a=0.25, c=0.0)

plt.figure(figsize=(7, 7))

# Nombre de pétales / branches (plus tu augmentes, plus c'est "mandala")
N = 12
angles = np.linspace(0, 2*np.pi, N, endpoint=False)

for i, theta in enumerate(angles):
    xr, yr = rotation(x, y, theta)
    # Variation légère pour donner du vivant
    plt.plot(xr, yr, linewidth=1.6, alpha=0.85)

# Mise en forme artistique
plt.gca().set_aspect('equal', adjustable='box')
plt.axis('off')
plt.title("Rosace parabolique (branches orientées selon l'axe des abscisses)", pad=18)
plt.show()
```



<u>Algorithme N°6</u> « “Œil” parabolique »

Œil parabolique : on combine deux paraboles horizontales orientées en sens opposé. Leur rencontre délimite une zone fermée que l’on remplit pour faire apparaître une forme centrale en amande. Le contraste entre les courbes et la surface colorée donne une figure organique, proche d’un œil stylisé ou d’un vitrail géométrique.

```
# Algorithme N°6 (artistique) : "Œil" parabolique (2 branches opposées + remplissage)
import numpy as np
import matplotlib.pyplot as plt

def parabole_droite(y, a=0.18, c=0.0):
    return a * y**2 + c

def parabole_gauche(y, a=0.18, c=0.0):
    return -a * y**2 + c

# Domaine
y = np.linspace(-7, 7, 1500)

# Deux branches couchées opposées
x1 = parabole_droite(y, a=0.16, c=-2.2)   # branche droite décalée vers la gauche
x2 = parabole_gauche(y, a=0.16, c= 2.2)   # branche gauche décalée vers la droite

plt.figure(figsize=(8, 5))

# Contours
plt.plot(x1, y, linewidth=2.2)
plt.plot(x2, y, linewidth=2.2)

# Remplissage (effet "vitrail / amande")
plt.fill_betweenx(y, x1, x2, alpha=0.35)

# Option : "pupille" géométrique au centre (cercle)
t = np.linspace(0, 2*np.pi, 600)
R = 1.1
plt.plot(R*np.cos(t), R*np.sin(t), linewidth=2.0, alpha=0.9)
plt.fill(R*np.cos(t), R*np.sin(t), alpha=0.25)

# Mise en forme artistique
plt.gca().set_aspect('equal', adjustable='box')
plt.axis('off')
plt.title("Forme artistique : Œil parabolique (branches orientées selon l'axe des abscisses)",
pad=18)
plt.show()
```