Curs 04

(*plan de curs*)

1. Argumentul unui număr complex, unghiul a două laturi

```
    # redenumiri de functii:
    rho = abs
    theta = cmath.phase
    fromRhoTheta = cmath.rect
    wait = pygame.time.wait
```

Semiplane, interiorul unui poligon convex, interiorul unei curbe Jordan.

2. Transformări geometrice.

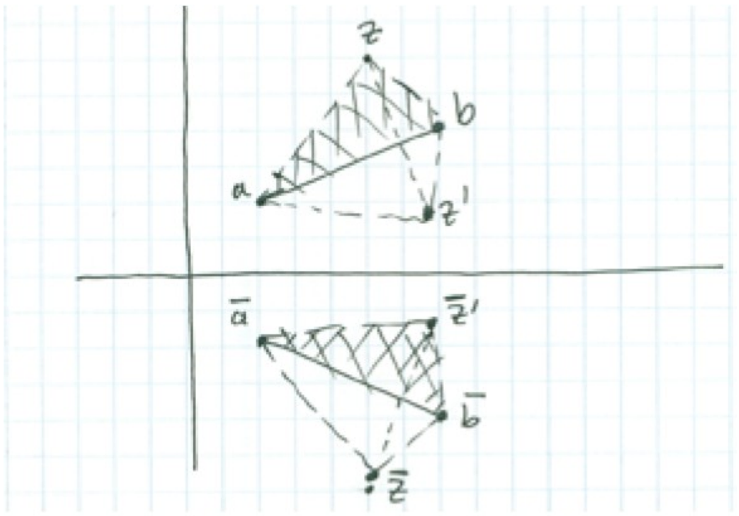
a) Translaţia $z' = z'_0 + (z - z_0)$

b) Omotetia $z' = z_0 + \lambda(z - z_0)$, cu λ real.

c) Rotaţia $z' = z_0 + \omega(z - z_0)$ cu $|\omega| = 1$.

d) Simetria faţă de un punct $z' = 2z_0 - z$.

e) Simetria faţă de dreapta *ab* $z' = a + \omega(\bar{z} - \bar{a})$ cu $\omega = \dfrac{b-a}{\bar{b}-\bar{a}}$.



f) Asemănarea

directă: $z' = a' + \omega(z - a)$ cu $\omega = \dfrac{b' - a'}{b - a}$.

inversă: $z' = a' + \omega(\bar{z} - \bar{a})$ cu $\omega = \dfrac{b' - a'}{\bar{b} - \bar{a}}$.

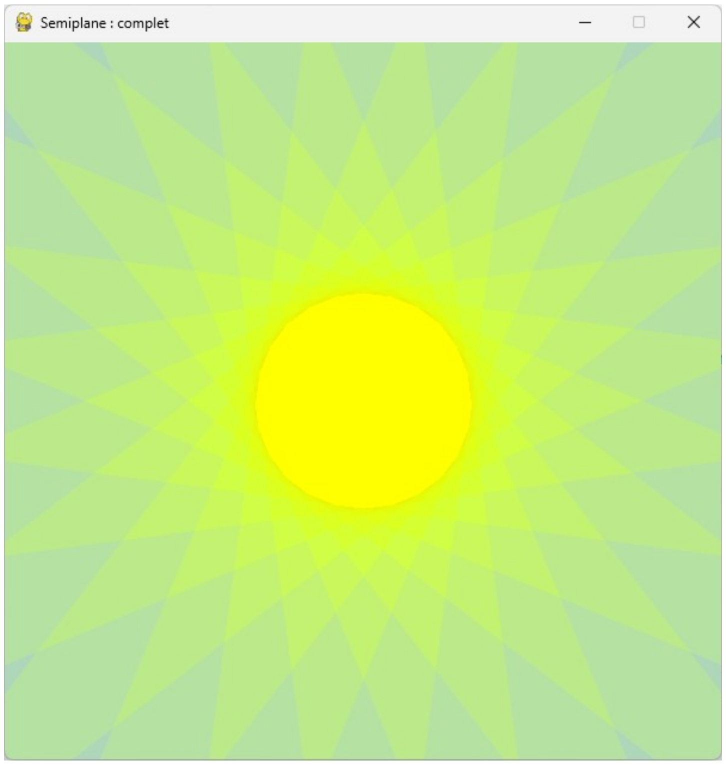
3. Topologia planului complex. Şirul puterilor unui număr complex, seria geometrică, curbe plane.

```
import ComplexPygame as C
import Color
import math

def Semiplane():
    def EsteInStanga(a, b, z):
        # return C.theta((z - a) / (b - a)) >= 0
        # return ((z - a) / (b - a)).imag >= 0
        return ((z - a) * (b - a).conjugate()).imag >= 0

    def ColoreazaSemiplane(p):
        # p este o lista circulara
        nrLaturi = len(p) - 1
        for coloana in C.screenColumns():
            for z in coloana:
                niv = 0
                for k in range(nrLaturi):
                    if EsteInStanga(p[k], p[k + 1], z):
                        niv += 1
                C.setPixel(z, Color.Index(10 * niv))
            C.refreshScreen()

    C.setXminXmaxYminYmax(-10, 10, -10, 10)
    N = 24
    delta = 2 * math.pi / N
    a = 5
    b = 3
    p = [C.fromRhoTheta(3, k * delta) for k in range(N + 1)]
    ColoreazaSemiplane(p)
    C.fillNgon(p, Color.Yellow)
```

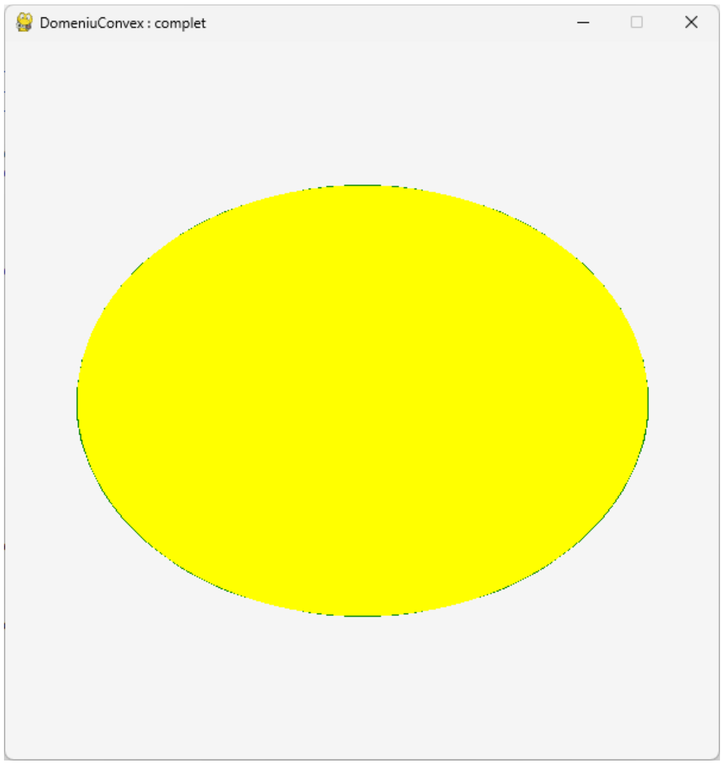


```
def DomeniuConvex():
    def EsteInInteriorConvex(p, z):
        # p este o lista circulara
        nrLaturi = len(p) - 1
        semn = ((z - p[0]) * (p[1] - p[0]).conjugate()).imag
        for k in range(1, nrLaturi):
            if semn * ((z - p[k]) * (p[k + 1] - p[k]).conjugate()).imag <= 0:
                return False
        return True

    def UmpleInteriorConvex(p, color):
        for coloana in C.screenColumns():
            for z in coloana:
                if EsteInInteriorConvex(p, z):
                    C.setPixel(z, color)
            C.refreshScreen()

    C.setXminXmaxYminYmax(-10, 10, -10, 10)
    N = 100
    delta = 2 * math.pi / N
    a = 8
    b = 6
    p = [complex(a * math.cos(k * delta), b * math.sin(k * delta)) for k in range(N +
1)]

    C.fillNgon(p, Color.Green)
    C.refreshScreen()
    UmpleInteriorConvex(p, Color.Yellow)
    C.refreshScreen()
```

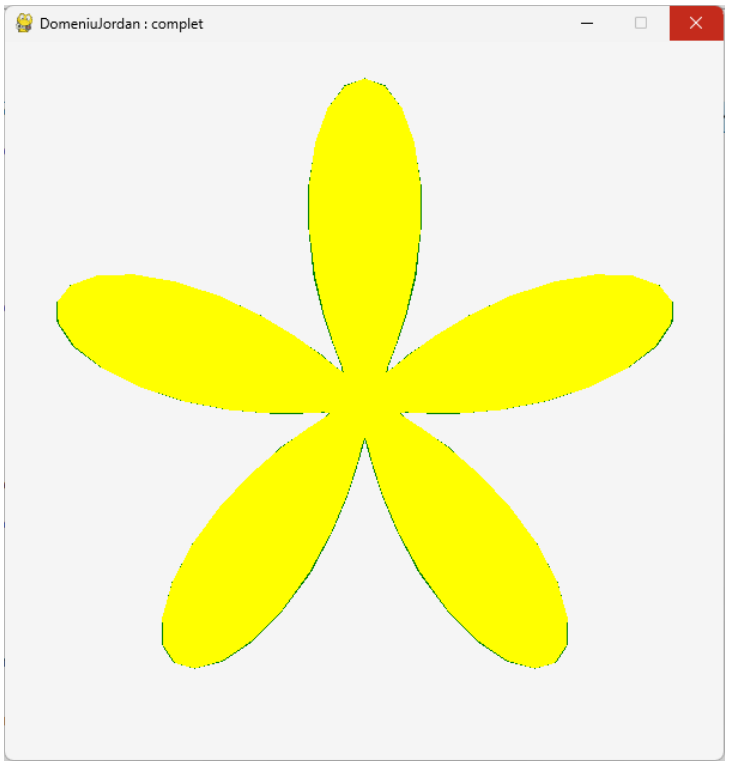


```
def DomeniuJordan():
    # p este o lista circulara
    def EsteInInteriorJordan(p, z):
        s = 0
        for k in range(1, len(p)):
            if z == p[k - 1]:
                continue
            s += C.theta((p[k] - z) / (p[k - 1] - z))
        return abs(s) > 0.1

    def UmpleInteriorJordan(p, col):
        for coloana in C.screenColumns():
            for z in coloana:
                if EsteInInteriorJordan(p, z):
                    C.setPixel(z, col)
            C.refreshScreen()

    C.setXminXmaxYminYmax(-10, 10, -10, 10)
    N = 100
    delta = 2 * math.pi / N
    p = []
    for k in range(N + 1):
        t = k * delta
        r = 5 + 4 * math.sin(5 * t)
        p.append(C.fromRhoTheta(r, t))

    C.fillNgon(p, Color.Green)
    C.refreshScreen()
    UmpleInteriorJordan(p, Color.Yellow)
    C.refreshScreen()
```



```python
if __name__ == '__main__':
    C.initPygame()

    C.run(Semiplane)
    C.run(DomeniuConvex)
    C.run(DomeniuJordan)
```