```
# Solve equations with sympy

# Bestimme die Gleichung der Geraden durch zwei Punkte in
# Normalvektorform:
def get_line_equation(p_1, p_2):
    "Return the equation of the line through p_1 and p_2"
    # sympy-Symbole müssen als solche "extra definiert" werden
    x_1,x_2= sp.symbols('x_1,x_2')
    normal_vector = get_normalvector(p_2-p_1)
    # sympy bietet _symbolische_ Gleichungen (und Lösungsmethoden
    # für diese):
    return sp.Eq(
        x_1*normal_vector[0]
        +x_2*normal_vector[1]
        -np.inner(normal_vector, p_1)
        ,0
    )

# Gleichung eines Kreises:
def get_circle_equation(p_m, radius):
    """Return the equation of the circle with center p_m and
    radius"""
    x_1,x_2= sp.symbols('x_1,x_2')
    return sp.Eq((x_1-p_m[0])**2+(x_2-p_m[1])**2-radius**2,0)

# Löse System von Gleichungen (in REELLEN Zahlen):
def get_solution_xy(equations):
    """Solve the system of equations, return LIST of solutions
    (maybe empty or singleton)"""
    x_1,x_2= sp.symbols('x_1,x_2')
    solutions = spsolvers.solve(
        equations, (x_1,x_2),
        domain=sp.S.Reals
    )
    if isinstance(solutions,list):
        # Leere Liste oder Liste mit mindestens 2 Lösungen
        return [npp(p) for p in solutions]
    # Otherwise it should hold: type(solutions) == dict:
    # "Eine" Lösung, eventuell in Form einer Gleichung: dict
    if x_1 in solutions and x_2 in solutions:
        return [npp(solutions[x_1],solutions[x_2])]
    print("!!! get_solution_xy returns equation!!!")
    return sp.Eq(solutions[x_1],solutions[x_2])

def get_intersection(obj_a, obj_b):
    """Find points of intersection for two geometrical objects"""
    return get_solution_xy([obj_a.equation, obj_b.equation])
```