```
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
import math


class wavefront:
    #lambdas for each of the zernike modes
    zernikefunctions = {
        1 : lambda rho, theta: 1,
        2 : lambda rho, theta: 2*rho*math.cos(theta),
        3 : lambda rho, theta: 2*rho*math.sin(theta),
        4 : lambda rho, theta: (3**0.5)*(2*rho*rho-1),
        5 : lambda rho, theta: (6**0.5)*rho*rho*math.sin(2*theta),
        6 : lambda rho, theta: (6**0.5)*rho*rho*math.cos(2*theta),
        7 : lambda rho, theta: (8**0.5)*(3*rho*rho*rho - 2*rho)*math.sin(theta
),
        8 : lambda rho, theta: (8**0.5)*(3*rho*rho*rho - 2*rho)*math.cos(theta
),
        9 : lambda rho, theta: (8**0.5)*rho*rho*rho*math.sin(3*theta),
        10: lambda rho, theta: (8**0.5)*rho*rho*rho*math.cos(3*theta),
        11: lambda rho, theta: (5**0.5)*(6*rho**4 - 6*rho*rho + 1),
        12: lambda rho, theta: (10**0.5)*(4*rho**4 - 3*rho*rho)*math.cos(2*the
ta),
        13: lambda rho, theta: (10**0.5)*(4*rho**4 - 3*rho*rho)*math.sin(2*the
ta),
        14: lambda rho, theta: (10**0.5)*(rho**4)*math.cos(4*theta),
        15: lambda rho, theta: (10**0.5)*(rho**4)*math.sin(4*theta)
    }

    def __init__(self, coefficients, x_res, y_res, radius):
        # Generate X Y coords
        X = np.arange(-1, 1, 2 / x_res)
        Y = np.arange(-1 * y_res / x_res, 1 * y_res / x_res, 2 / y_res)
        X, Y = np.meshgrid(X, Y)

        #Generate theta rho coords
        theta = np.arctan2(Y, X)
        rho = np.sqrt(X ** 2 + Y ** 2)

        #Build the wavefront using the provided coefficients
        Z = np.zeros(X.shape)
        for idx, coeff in enumerate(coefficients):
            Z += coeff * np.vectorize(self.zernikefunctions[idx + 1])(rho, the
ta)

        self.image = Z #* (rho < radius / x_res).astype(np.int)
        self.Xcoords = X
        self.Ycoords = Y
        self.x_res = x_res
        self.y_res = y_res
        self.radius = radius
        self.thetacoords = theta
```

```
        self.rhocoords = rho

    #Build a wavefront object from an amplitude map
    @classmethod
    def fromimage(cls, image):
        y_res = len(image)
        x_res = len(image[0])

        # Generate X Y coords
        X = np.arange(-1, 1, 2 / x_res)
        Y = np.arange(-1, 1, 2 / y_res)
        X, Y = np.meshgrid(X, Y)

        coefficients = [0 for i in range(15)]

        pix_count = 0

        for xr, yr, zr in zip(X, Y, image):
            for x, y, z in zip(xr, yr, zr):
                rho = math.sqrt(x ** 2 + y ** 2)
                theta = math.atan2(y, x)
                if rho > 1:
                    continue
                coefficients = [coefficients[i] + z * cls.zernikefunctions[i +
 1](rho, theta) for i in range(15)]
                pix_count += 1

        return [coef / pix_count for coef in coefficients]

#Previous implementation, left as a reference
"""
def zernike(coefficients, x, y):
    rho = math.sqrt(x**2 + y**2)
    theta = math.atan2(y, x)
    if rho > 1:
        return 0
    amplitude = 0
    for i, coef in enumerate(coefficients):
        amplitude += coef*zernikefunctions[i+1](rho, theta)
    return amplitude

def zernike_compute(coefficients, X, Y):
    Z = []
    for i in range(len(X)):
        xr = X[i]
        yr = Y[i]
        zr = []
        for j in range(len(xr)):
            x = xr[j]
            y = yr[j]
            zr.append(zernike(coefficients, x, y))
        Z.append(zr)
    return Z
```

```
"""


if __name__ == "__main__":

    #coefs = [0, 0.3, 0.3, 0.3, 0, 0.2, 0.1]
    coefs = [0,0,0,0,1]
    Z = wavefront(coefs, 200, 200, 100).image
    #coefs2 = map_to_zernike(Z)
    #print('Error = ', ['{:.2f}%'.format(100*math.fabs(c1-c2)/(math.fabs(c1) i
f c1 != 0 else 1)) for c1, c2 in zip(coefs, coefs2)])
    plt.figure()
    plt.subplot(1,3,1)
    plt.imshow(Z, cmap=cm.coolwarm)
    plt.subplot(1,3,2)
    plt.imshow(np.gradient(Z)[0], cmap = cm.coolwarm)
    plt.subplot(1,3,3)
    plt.imshow(np.gradient(np.gradient(Z)[0])[0], cmap = cm.coolwarm)
    plt.show()
```