```python
import numpy as N

def ApproximateJacobian(f, x, dx=1e-6):
    """Return an approximation of the Jacobian Df(x) as a numpy matrix"""
    try:
        n = len(x)
    except TypeError:
        n = 1
    fx = f(x)
    Df_x = N.matrix(N.zeros((n,n)))
    for i in range(n):
        v = N.matrix(N.zeros((n,1)))
        v[i,0] = dx
        Df_x[:,i] = f(x + v) - fx
    return Df_x

class Polynomial(object):
    """Callable polynomial object.

    Example usage: to construct the polynomial p(x) = x^2 + 2x + 3,
    and evaluate p(5):

    p = Polynomial([1, 2, 3])
    p(5)"""

    def __init__(self, coeffs):
        self._coeffs = coeffs

    def __repr__(self):
        return "Polynomial(%s)" % (", ".join([str(x) for x in self._coeffs]))

    def f(self,x):
        ans = self._coeffs[0]
        for c in self._coeffs[1:]:
            ans = x*ans + c
        return ans

    def __call__(self, x):
        return self.f(x)

```