Polynomials: The Make 'Em and Break 'Em Game:

Challenge:
Make the most ridiculously complicated polynomial by multiplying as many binomials as you can!
I will factor them on the spot.

For example:
Consider:

$$y = (x+3) * (x-5) * (x+1) * \ldots * (x-243) * (x+14.707) * \ldots$$

The roots of that polynomial are where y=0. These occur at x = {-3, +5, -1, …, +243, -14.707, …}

DO NOT TELL ME THE ROOTS YOU USED.
JUST TELL ME THE RESULTING COEFFICIENTS OF THE POLYNOMIAL.

numpy.roots

numpy.roots(*p*) [source]

Return the roots of a polynomial with coefficients given in p.

The values in the rank-1 array *p* are coefficients of a polynomial. If the length of *p* is n+1 then the polynomial is described by:

```
p[0] * x**n + p[1] * x**(n-1) + ... + p[n-1]*x + p[n]
```

Parameters : **p** : *array_like*

Rank-1 array of polynomial coefficients.

Returns : **out** : *ndarray*

An array containing the complex roots of the polynomial.

Raises : **ValueError**

When *p* cannot be converted to a rank-1 array.

See also:

poly	Find the coefficients of a polynomial with a given sequence of roots.
polyval	Evaluate a polynomial at a point.
polyfit	Least squares polynomial fit.
poly1d	A one-dimensional polynomial class.

Notes

The algorithm relies on computing the eigenvalues of the companion matrix [R241].

References

[R241] *(1, 2)* R. A. Horn & C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1999, pp. 146-7.

Examples

```
>>> coeff = [3.2, 2, 1]
>>> np.roots(coeff)
array([-0.3125+0.46351241j, -0.3125-0.46351241j])
```