In [ ]:
```
# Degree first, then down
x_vals = np.linspace(1, 5, 1000)
poly_approx = np.polyval(ys[::-1][1:], x_vals)
```

In [ ]:
```
# Exercise 20, part 2
# Getting random symmetric matrix
size = 1000
Q = ortho_group.rvs(size)
Evals = np.random.uniform(1, 5, size)
R = np.diag(Evals)
Mat = Q @ R @ Q.T
b = np.random.uniform(0, 1, size)
Actual_sol = sp.linalg.solve(Mat,b, assume_a='pos')
```

In [ ]:
```
# Estimating the Soln of the system with my poly_approx
# Backward
Backward_error = np.linalg.norm(poly_approx - Actual_sol, ord = 2)/ np.linalg.norm(Actual_sol, ord=2)
Forward_error =  np.linalg.norm(Mat @ poly_approx - b, ord=2) / np.linalg.norm(b, ord = 2)
```

In [ ]:
```
print("Backward_error: ", Backward_error )
print("Forward_error: ", Forward_error)
```

In [ ]:
```
plt.plot(Actual_sol)
```