Q Evaluator

Functions for evaluating the Q polynomials at specific points. Usage:

$$x^2 = 9240 = \sum_{x=0}^{\infty} \frac{1}{x}$$

```
import itertools
```

```
# transpose the csv, to put the polynomials in the rows, with coefficients spread across the
import csv
a = zip(*csv.reader(open("data/q_to_denom_1000.csv", "r")))
csv.writer(open("data/q_to_denom_1000_T.csv", "w")).writerows(a)
```

```
import pandas as pd
polys = pd.read_csv('data/q_to_denom_500.csv',dtype=str)
polys.rename(columns={'Unnamed: 0':'p/q'},inplace=True)
polys.set_index('p/q', inplace=True)
```

```
def eval_q_at_x(fraction, x):
    """
    Retrieves the Q polynomial of the specified fraction. Evaluates the polynomial at the g
    The fraction should be passed as an array.
    """
    string_fraction = f"[{fraction[0]},{fraction[1]}]"
    p = polys.loc[string_fraction].to_list()
    p = [int(s) for s in p]
    # cuts off leading zeros
    i = 0
    sum = p[i]
    while sum == 0:
        i += 1
        sum += abs(p[i])
    p = p[i:]
    # evaluates polynomial
    val = 0
    for i, coeff in enumerate(p[::-1]):
        val += x**i * coeff
    return val
```

```
polys
```

```
          0  1  2  3  4  5  6  7  8  9  ...  489  490  \
p/q                                      ...
[0,1]     0  0  0  0  0  0  0  0  0  0  ...    0    0
[1,1]     0  0  0  0  0  0  0  0  0  0  ...    0    0
[1,0]     0  0  0  0  0  0  0  0  0  0  ...    0    0
[1,2]     0  0  0  0  0  0  0  0  0  0  ...    0    0
```

```
[1,3]      0  0  0  0  0  0  0  0  0  0  ...              0            0
...       .. .. .. .. .. .. .. .. .. ..  ...            ...          ...
[344,763]  0  0  0  0  0  0  0  0  0  0  ... -1462827789230  120884386754
[346,767]  0  0  0  0  0  0  0  0  0  0  ... -1481074898910  122131641490
[374,829]  0  0  0  0  0  0  0  0  0  0  ... -2670388590762  207465078342
[379,828]  0  0  0  0  0  0  0  0  0  0  ... -1261547183228  103180046874
[391,852]  0  0  0  0  0  0  0  0  0  0  ... -1356316751766  109613593005

                   491         492         493      494      495   496  497 498
p/q
[0,1]                0           0           0        0        0     0    0   1
[1,1]                0           0           0        0        0     0    0   1
[1,0]                0           0           0        0        0     0    0   0
[1,2]                0           0           0        0        0     0    0   1
[1,3]                0           0           0        0        0     0   -1   1
...                ...         ...         ...      ...      ...   ...  ...  ..
[344,763]  -9142412360   624620440  -37889915  1992227   -87625  3044  -75   1
[346,767]  -9218104715   628601852  -38065090  1998314   -87775  3046  -75   1
[374,829] -14741458488   945480381  -53788671  2649146  -108972  3533  -81   1
[379,828]  -7752093156   528428000  -32146254  1705647   -76300  2724  -70   1
[391,852]  -8141976244   549042972  -33065634  1738389   -77140  2736  -70   1

[75919 rows x 499 columns]
```

```python
import numpy as np
import matplotlib.pyplot as plt
def down_the_triangle(alpha, gamma, x):
    alpha = np.array(alpha)
    gamma = np.array(gamma)
    # print recurrence matrix for gamma
    R = np.array(
    [[1,0],
    [(-1)**(gamma[0]+1)* x**gamma[1], eval_q_at_x(gamma, x)]]
    )
    print(R)
    # print x evaluated at the polynomials alpha +n gamma
    seq = []
    for i in range(20):
        mix = alpha + gamma*i
        seq.append(eval_q_at_x(mix,x))
        print(seq[-1])
    # plot all of these points
    reals = [s.real for s in seq]
    imags = [s.imag for s in seq]
    colors = np.arange(len(seq))
    plt.scatter(reals,imags, c= colors)
```

```
# from 0 to 1, the Fibonacci sequence
down_the_triangle([0,1],[1,1],1)
```

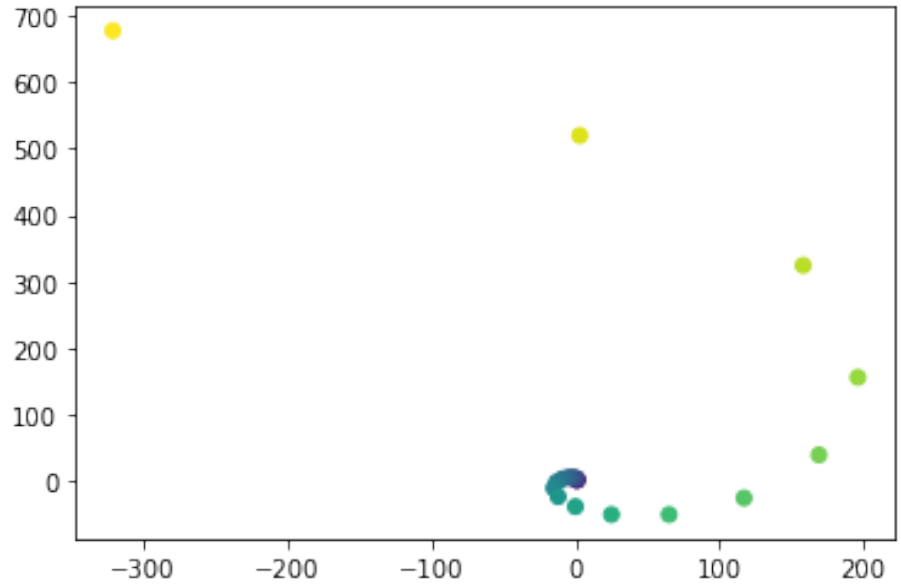
```
[[1 0]
 [1 1]]
1
1

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/opt/anaconda3/envs/rutabaga/lib/python3.8/site-packages/pandas/core/indexes/base.py in get
   3079             try:
-> 3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item

KeyError: '[2,3]'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-100-6489cf6f2997> in <module>
      1 # from 0 to 1, the Fibonacci sequence
----> 2 down_the_triangle([0,1],[1,1],1)

<ipython-input-28-ebaf7035accb> in down_the_triangle(alpha, gamma, x)
     14     for i in range(20):
     15         mix = alpha + gamma*i
---> 16         seq.append(eval_q_at_x(mix,x))
     17         print(seq[-1])
     18     # plot all of these points

<ipython-input-27-d865e257fda8> in eval_q_at_x(fraction, x)
      5     """
      6     string_fraction = f"[{fraction[0]},{fraction[1]}]"
----> 7     p = polys.loc[string_fraction].to_list()
      8     p = [int(s) for s in p]
      9     # cuts off leading zeros

~/opt/anaconda3/envs/rutabaga/lib/python3.8/site-packages/pandas/core/indexing.py in __getit
```

```
    893
    894             maybe_callable = com.apply_if_callable(key, self.obj)
--> 895             return self._getitem_axis(maybe_callable, axis=axis)
    896
    897     def _is_scalar_access(self, key: Tuple):

~/opt/anaconda3/envs/rutabaga/lib/python3.8/site-packages/pandas/core/indexing.py in _getite
   1122         # fall thru to straight lookup
   1123         self._validate_key(key, axis)
-> 1124         return self._get_label(key, axis=axis)
   1125
   1126     def _get_slice_axis(self, slice_obj: slice, axis: int):

~/opt/anaconda3/envs/rutabaga/lib/python3.8/site-packages/pandas/core/indexing.py in _get_la
   1071     def _get_label(self, label, axis: int):
   1072         # GH#5667 this will fail if the label is not present in the axis.
-> 1073         return self.obj.xs(label, axis=axis)
   1074
   1075     def _handle_lowerdim_multi_index_axis0(self, tup: Tuple):

~/opt/anaconda3/envs/rutabaga/lib/python3.8/site-packages/pandas/core/generic.py in xs(self,
   3736                 raise TypeError(f"Expected label or tuple of labels, got {key}") fro
   3737         else:
-> 3738             loc = index.get_loc(key)
   3739
   3740             if isinstance(loc, np.ndarray):

~/opt/anaconda3/envs/rutabaga/lib/python3.8/site-packages/pandas/core/indexes/base.py in get
   3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:
-> 3082                 raise KeyError(key) from err
   3083
   3084         if tolerance is not None:

KeyError: '[2,3]'
```

```
down_the_triangle([0,1],[1,1],0+1j)
```

```
[[1.+0.j 0.+0.j]
 [0.+1.j 1.+0.j]]
(1+0j)
(1+0j)
(1+1j)
(1+2j)
3j
(-2+4j)
(-5+4j)
```

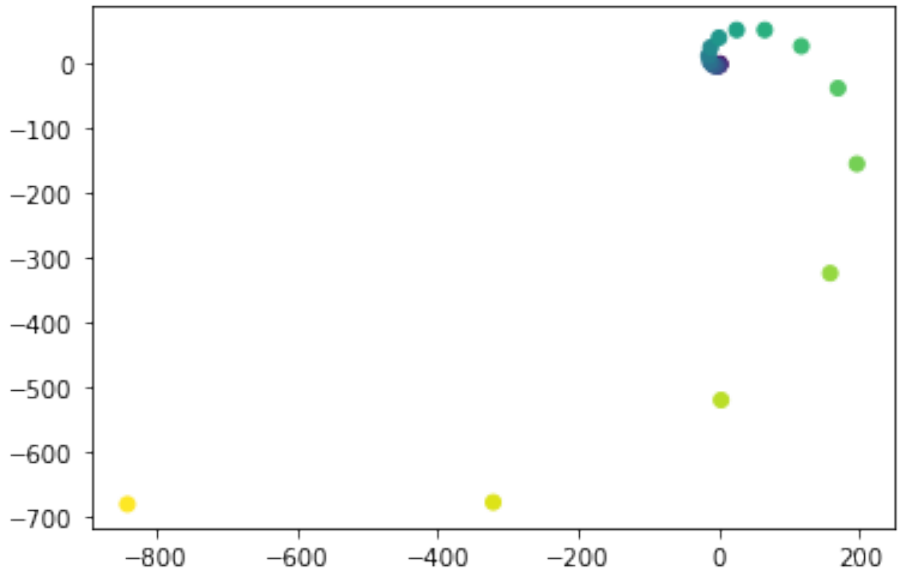
```
(-9+2j)
(-13-3j)
(-15-12j)
(-12-25j)
-40j
(25-52j)
(65-52j)
(117-27j)
(169+38j)
(196+155j)
(158+324j)
(3+520j)
(-321+678j)
```




```
down_the_triangle([1,2],[0,1],0+1j)
```

```
[[ 1.+0.j  0.+0.j]
 [-0.-1.j  1.+0.j]]
(1+0j)
(1-1j)
(1-2j)
-3j
(-2-4j)
(-5-4j)
(-9-2j)
(-13+3j)
(-15+12j)
```

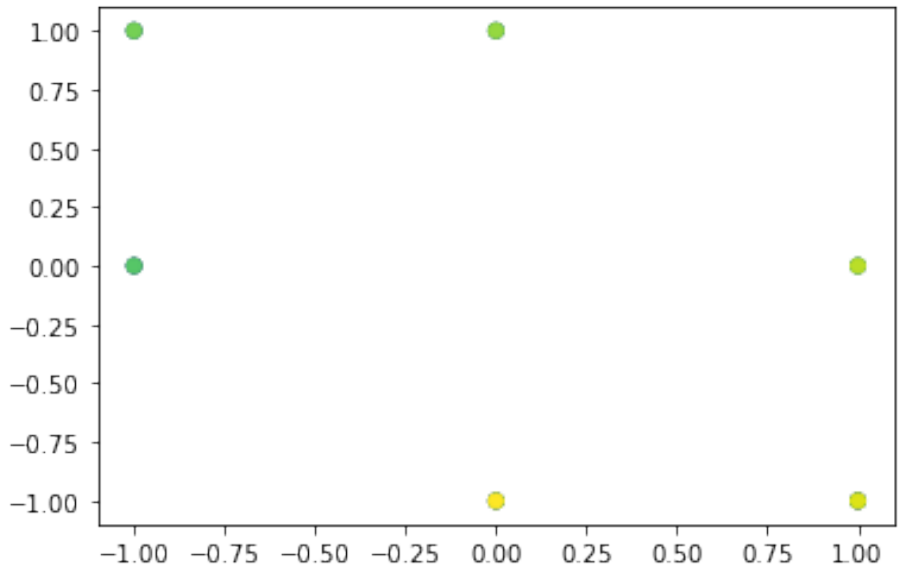
```
(-12+25j)
40j
(25+52j)
(65+52j)
(117+27j)
(169-38j)
(196-155j)
(158-324j)
(3-520j)
(-321-678j)
(-841-681j)
```




```
down_the_triangle([1,3],[1,2],0+1j)
```

```
[[ 1.+0.j  0.+0.j]
 [-1.+0.j  1.+0.j]]
(1-1j)
-1j
(-1+0j)
(-1+1j)
1j
(1+0j)
(1-1j)
-1j
(-1+0j)
(-1+1j)
1j
```

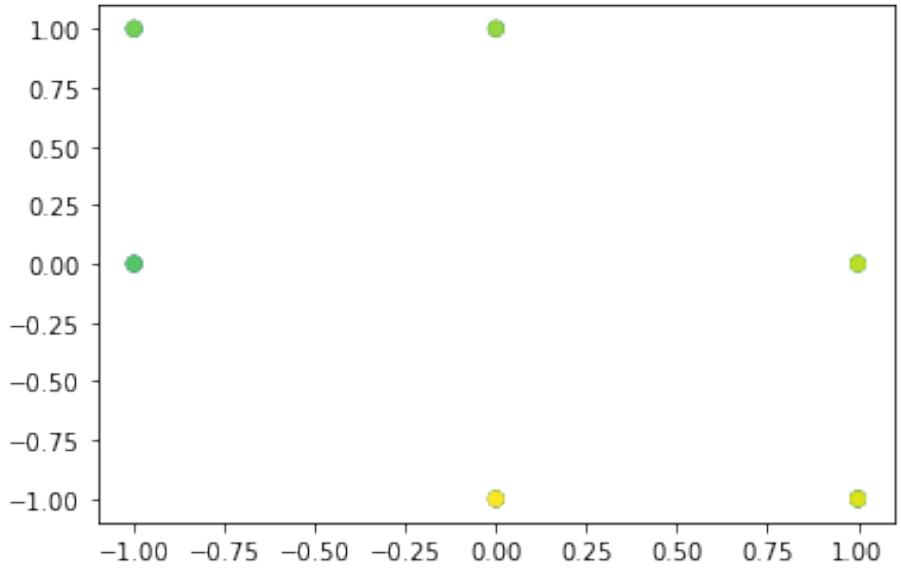
```
(1+0j)
(1-1j)
-1j
(-1+0j)
(-1+1j)
1j
(1+0j)
(1-1j)
-1j
```




```
down_the_triangle([1,3],[1,2],0+1j)
```

```
[[ 1.+0.j  0.+0.j]
 [-1.+0.j  1.+0.j]]
(1-1j)
-1j
(-1+0j)
(-1+1j)
1j
(1+0j)
(1-1j)
-1j
(-1+0j)
(-1+1j)
1j
(1+0j)
(1-1j)
```

```
-1j
(-1+0j)
(-1+1j)
1j
(1+0j)
(1-1j)
-1j
```




```
import math
math.gcd(12,60)
```

```
12
```

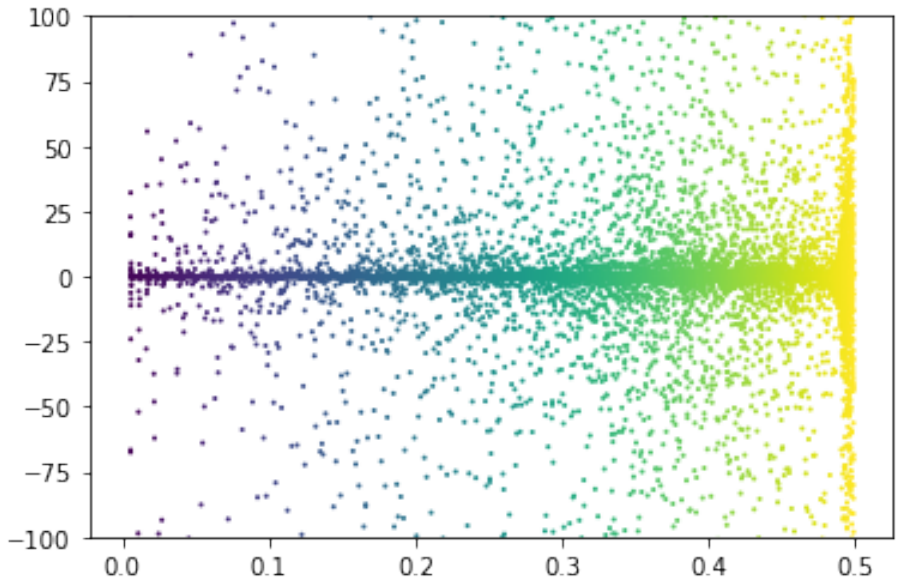
```
polys.index
```

```
Index(['[0,1]', '[1,1]', '[1,0]', '[1,2]', '[1,3]', '[1,4]', '[1,5]', '[1,6]',
       '[1,7]', '[1,8]',
       ...
       '[309,688]', '[327,728]', '[358,797]', '[339,752]', '[321,712]',
       '[344,763]', '[346,767]', '[374,829]', '[379,828]', '[391,852]'],
      dtype='object', name='p/q', length=75919)
```

```
# plot of the antiroot function
# get all fractions with denom up to max denom in unit interval to 1/2
import math
x = 0.25 + 0.25j
# get all of the fracs available, and sort them
def string_to_list(x):
    return list(map(int,x[1:-1].split(',')))
```

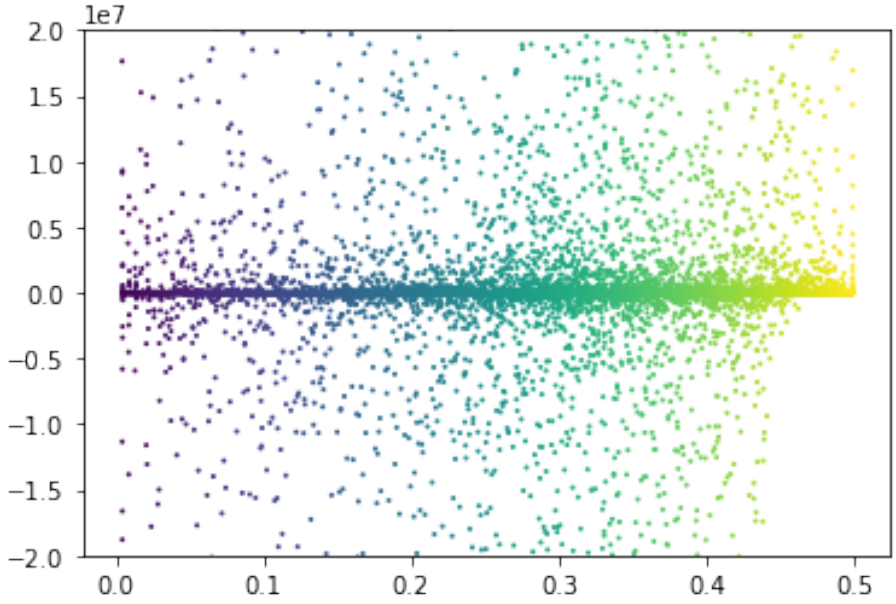
```
fracs = [string_to_list(i) for i in polys.index[3:]] # to avoid division by zero
fracs = sorted(fracs,key=lambda s:  s[0]/s[1])
# print(fracs)

ys = []
frac_points = [s[0]/s[1] for s in fracs]
for f in fracs:
    ys.append(eval_q_at_x(f,x))
# plot all of these points
reals = [s.real for s in ys]
imags = [s.imag for s in ys]
colors = np.arange(len(ys))
plt.scatter(frac_points,reals, c=colors, s=1)
# plt.xlim(-1,2)
plt.ylim(-100,100)
```

```
(-100.0, 100.0)
```



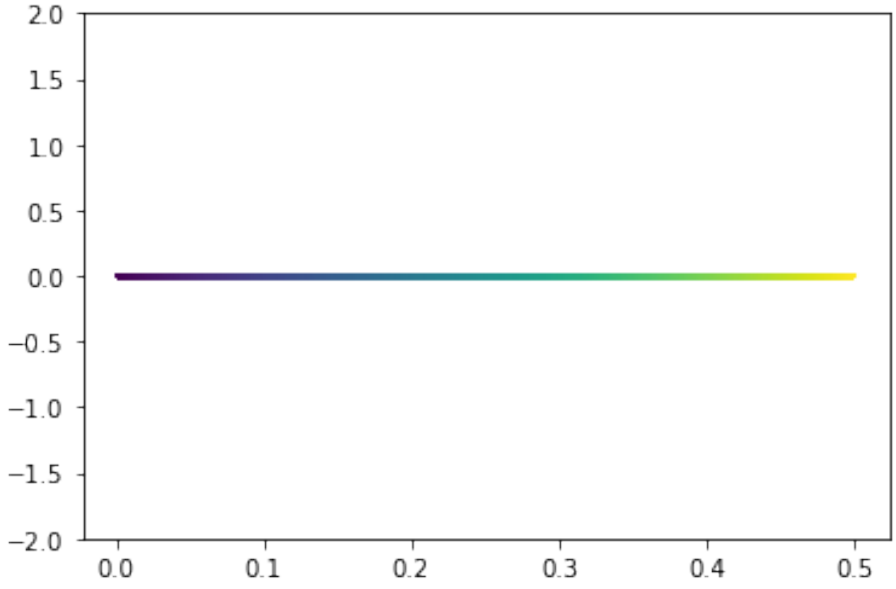

```
plt.scatter(frac_points, reals, c= colors, s=1)
# plt.xlim(-1,2)
plt.ylim(-20000000,20000000)
```

```
(-20000000.0, 20000000.0)
```



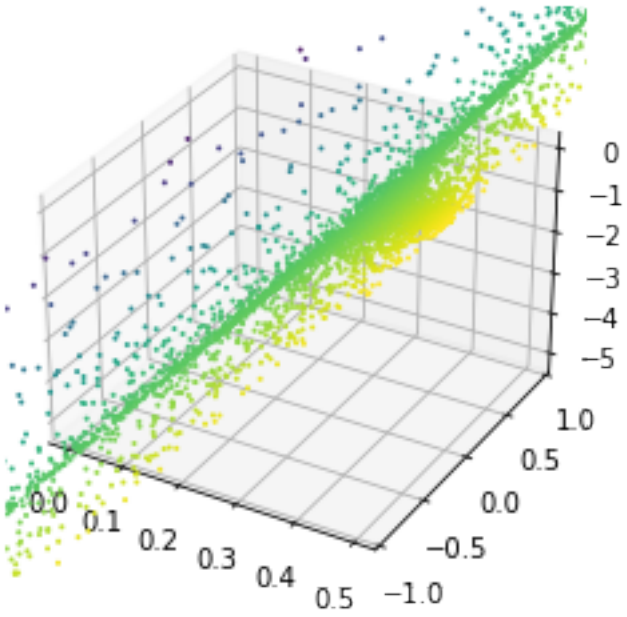

```
plt.scatter(frac_points, imags, c= colors, s=1)
# plt.xlim(-1,2)
plt.ylim(-2,2)
```

```
(-2.0, 2.0)
```



```
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.scatter(frac_points,reals,imags,c=colors, s = 1)
plt.ylim(-1,1)
```

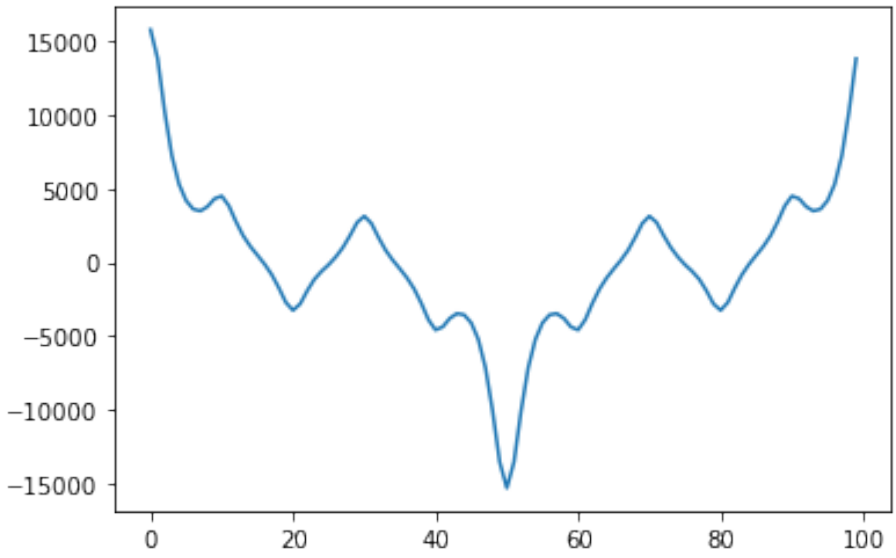
```
(-1.0, 1.0)
```




```
plt.plot(np.fft.fft(reals))
```

```
/Users/adjourner/opt/anaconda3/envs/rutabaga/lib/python3.8/site-packages/numpy/core/_asarray
  return array(a, dtype, copy=False, order=order)
```

```
[<matplotlib.lines.Line2D at 0x7faad2456e20>]
```



ys

```
[(1+0j),
 (0.001868909341328247-0.0030284838867021442j),
 (0.00797093107579272+0.002097875683091388j),
 (-0.0010534890529422876-0.0010244735773182224j),
 (-0.06271117925643921-0.05475044250488281j),
 (0.0911865234375+0.09578704833984375j),
 (-0.0050578830456196044+0.0029108248008355986j),
 (5.480170810973141e-05+5.798541926322957e-05j),
 (0.023100435733795166-0.06761550903320312j),
 (2.103568914242865e-06-1.2142143002961352e-06j),
 (0.17578125+0.28125j),
 (1.5749033518833994e-05-1.2009276820073782e-05j),
 (0.06531006097793579-0.013017654418945312j),
 (7.5332619318861096e-06+2.7857008480550105e-06j),
 (0.0009287568902074383+0.0007721307238739428j),
 (-0.067657470703125-0.01354217529296875j),
 (-0.007359921932220459+0.046749114990234375j),
 (-8.426689130986215e-06+1.2851776059669358e-05j),
 (-0.0042045207939587215-0.005119983406984829j),
 (-3.5220776265303724e-05+0.00011319738983326809j),
 (-0.3125-0.5j),
 (-3.6605319282019133e-06+7.956979354103233e-05j),
 (-0.0034002488616202697-0.002501420843145752j),
 (8.36566758751053e-06+2.25572692842723e-06j),
 (0.005644381046295166+0.046642303466796875j),
```

```
-0.5j,
(-0.00022958755124236063-0.00015818313841009513j),
(-1.9675956606590973e-09-2.4757845553928566e-09j),
(-0.002973020076751709+0.0022220611572265625j),
(5.154642328656272e-13+1.1758803604081825e-13j),
(0.01953125-0.03125j),
(-3.4280512812033e-11+3.9677007758318734e-11j),
(0.0016713738441467285-0.0033998489379882812j),
(2.9901930434545377e-09+2.789235959414225e-09j),
(8.113604863879686e-05+0.00015054962858584986j),
(0.03900146484375+0.01354217529296875j),
(0.025036394596099854+0.009016990661621094j),
(1.2017840565923435e-06+4.3378271850673664e-07j),
(0.0030287547188585506-0.0010996943452710184j),
(9.29762512982236e-05-0.00015029426362351268j),
(0.6875+0j),
(0.00758285275776242-0.003539653692226163j),
(0.15671739707528687+0.03998570057262896j),
(0.007590458196665415+0.08482051912842886j),
(0.6934018731117249+0.4774770736694336j),
(0.96478271484375+0.7449874877929688j),
(-1.7723821604233159+3.227056080742841j),
(20.58855730166363-65.87412682003018j),
(0.43139129877090454+4.307491302490234j),
(1593.9756377091264+808.3223605726108j),
(1+0j),
(4463.563157214131+10481.314209464244j),
(-2.700042188167572+10.590886116027832j),
(12079.313383316085+13175.773765394877j),
(-141.59104723807144-48.42737866456707j),
(1.663787841796875+7.570930480957031j),
(-1.0788435339927673+13.664430618286133j),
(44815.5115022184+6244.344209541564j),
(-233.24717653594024-3.078198326316169j),
(64653.80039561372-4740.745641299023j),
(1.6875+0.5j),
(78598.23601384224-56312.92853235931j),
(-299.00190751011877+177.61455607551284j),
(6567.396202359893-154577.833337293j),
(10.1173539775772858+18.648515701293945j),
(8.749725341796875+8.443000793457031j),
(-70.46574050286162+586.1602091118757j),
(-510561.0667559749-42147.33077510116j),
(18.972483217716217+24.74915885925293j),
(-988982.848985193-1438613.8951929575j),
(4.01953125+1.96875j),
```

```
(1699190.3331366326-5730005.443950382j),
(12.282791674137115+56.735724449157715j),
(19901804.00353418-9281314.265748864j),
(-6512.807115451067+67.63071256868272j),
(2+0.5j),
(-12.534602582454681+108.95229053497314j),
(221437415.5991234+127222486.80622031j),
(-20344.083051355803-6119.713379152622j),
(639315697.7654868+482482440.16351163j),
(2.6875+1j),
(1551495860.594794+1968395423.0826688j),
(-56712.28568613199-34692.632724133335j),
(2139423103.4213228+7498488704.207471j),
(-122.3648676276207+320.69659900665283j),
(-0.90142822265625+119.93883514404297j),
(-133874.30917296352-162368.1776313139j),
(-27341581342.026566+73626265654.36612j),
(-304.80100959539413+529.7769041061401j),
(-155769412430.39282+193485778158.00287j),
(9.42578125+11.15625j),
(-660656931456.9808+424486447874.5831j),
(-685.9331621527672+843.3214311599731j),
(-2406401724897.6597+612797203740.8263j),
(-92932.44993017316-2099330.96852027j),
(-122.42648315429688+359.01700592041016j),
(-1444.2505030035973+1284.909945487976j),
(-22927999202974.617-9536040264939.363j),
(1820533.636976443-6390925.358761552j),
(-59247957343984.516-51536119997792.71j)]
```