```
#! /opt/local/bin/python
#!
/home/dantopa/spacktivity/ubuntu-22.04-dantopa-docker-spack/opt/spack/linux-ubuntu22.04-haswell
gcc-12.2.0/python-3.10.6-ybcookynohghdr6i72gwalqle6hq27z5/bin/python
# Solving Bevington example 6.1 with Python
from datetime import date
today = date.today( )
print( today.strftime("%b-%d-%Y") )
import os
print(os.path.dirname(os.path.realpath(__file__)))

print( "importing numpy..." )
import numpy as np

# input data
A = np.array([ [1, 1], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [1, 8], [1, 9] ])
print( "design matrix A\n", np.matrix( A ) )
T = np.array([15.6, 17.5, 36.6, 43.8, 58.2, 61.6, 64.2, 70.4, 98.8])
print ( "data vector T =", np.matrix( T ))

# set up normal equations
W = np.dot( A.transpose(), A )
print( "Gram matrix A*A\n", np.matrix( W ) )
b = np.dot( A.transpose(), T )

# solve normal equations
xls = np.linalg.solve( W, b )
print ( "least squares solution xls =", xls )

# pseudoinverse solution
Winv  = np.linalg.inv( W )
Ap    = np.dot( Winv, A.transpose( ) )
xpinv = np.dot( Ap, T.transpose( ) )
print( "pseudoinverse        xpinv =", xpinv )

# error propagation
residual = np.dot( A, xls ) - T
t2 = np.dot( residual, residual )
print( "least total squared error =", t2 )
[ m, n ] = A.shape
print( "A matrix dimensions =", [ m, n ] )
diag  = np.diag( Winv )
sigma = np.sqrt( t2 / ( m - n ) * diag )
print ( "sigma =", sigma )

# numerical errors
print( "\nnumerical errors\n" )
xlsExact = [ 1733 / 360, 1129 / 120 ]
numericErrorXLS = xls - xlsExact
print( "numeric errors, fit parameters" )
print( numericErrorXLS )

sigmaExact = np.sqrt( np.array( [ 108297055, 3419907 ] ) / 35 ) / 360
numericErrorSigma = sigma - sigmaExact
print( "\nnumeric errors, sigma parameters" )
print( numericErrorSigma )


# machine epsilon
machineEpsilon = np.finfo(float).eps
print( "\nmachine epsilon =", np.finfo(float).eps )
```

```
60 numericErrorXLSeps   = numericErrorXLS   / machineEpsilon
61 numericErrorXLSsigma = numericErrorSigma / machineEpsilon
62 print( "\nnumeric epsilons, fit parameters:   ", numericErrorXLSeps )
63 print( "\nnumeric epsilons, sigma parameters: ", numericErrorXLSsigma )
64
65 # dantopa@Quaxolotl.attlocal.net:least-squares $ ./least-squares.py
66 # Sep-05-2022
67 # /Volumes/T7-Touch/repos/github/jop/python/genesis/least-squares
68
69 # importing numpy...
70 # design matrix A
71 #  [[1 1]
72 #  [1 2]
73 #  [1 3]
74 #  [1 4]
75 #  [1 5]
76 #  [1 6]
77 #  [1 7]
78 #  [1 8]
79 #  [1 9]]
80 # data vector T = [[15.6 17.5 36.6 43.8 58.2 61.6 64.2 70.4 98.8]]
81 # Gram matrix A*A
82 #  [[  9  45]
83 #  [ 45 285]]
84 # least squares solution xls = [4.81388889 9.40833333]
85 # pseudoinverse        xpinv = [4.81388889 9.40833333]
86 # least total squared error = 316.6580555555554
87 # A matrix dimensions = [9, 2]
88 # sigma = [4.88620631 0.86830165]
89 #
90 # numerical errors
91 #
92 # numeric errors, fit parameters
93 # [ 9.76996262e-15 -1.77635684e-15]
94 #
95 # numeric errors, sigma parameters
96 # [-8.88178420e-16 -2.22044605e-16]
97 #
98 # machine epsilon = 2.220446049250313e-16
99 #
100 # numeric epsilons, fit parameters:    [44. -8.]
101 #
102 # numeric epsilons, sigma parameters:  [-4. -1.]
103
```