Load the coefficients of the pure state (and verify that the norm is one):

```python
In [1]: import numpy as np

Load the coefficients of the pure state (and verify that the norm is one):

```python
In [2]: A = np.loadtxt('pset2.txt', dtype=np.complex128)
np.linalg.norm(A)

Out[2]: 1.0

(a) Schmidt coefficients: They are the singular values of $A$.

```python
In [3]: def schmidt(A):
    ...:     return np.linalg.svd(A)[1]
    ...:     # ALTERNATIVE: return np.sqrt(np.linalg.eigvalsh(A.conj().transpose() @ A))

schmidt(A)

Out[3]: array([0.94455344, 0.32835772])

(b) Partial traces: We use the formulas from Problem 1 on Exercise Set 2:

```python
In [4]: rho_X = A @ A.conj().transpose()
 rho_X

Out[4]: array([[0.57203512+0.j        , 0.03479877+0.38393501j],
                   [0.03479877-0.38393501j, 0.42796488+0.j        ]])

In [5]: rho_Y = A.transpose() @ A.conj()
 rho_Y

Out[5]: array([[0.11283263+0.j        , 0.03629507-0.05089396j],
                   [0.03629507+0.05089396j, 0.88716737+0.j        ]])

(c) Trace distance: The following implementation works for arbitrary operators $X$ and $Y$, not just Hermitian ones.

```python
In [6]: def trace_dist(X, Y):
    ...:     return np.sum(np.linalg.svd(X - Y)[1])

trace_dist(rho_X, rho_Y)

Out[6]: 1.2648247260172776

(d) Fidelity: Let’s use the formula

$$
F(\rho, \sigma) = \text{tr}[\sqrt{\sqrt{\rho} \sigma \sqrt{\rho}}].
$$
```python
from scipy.linalg import sqrtm

def fidelity(rho, sigma):
    sqrt_rho = sqrtm(rho)
    f = np.trace(sqrtm(sqrt_rho @ sigma @ sqrt_rho))
    assert np.isclose(f, np.real(f))
    return np.real(f)

fidelity(rho_X, rho_Y)
```

```
0.7746319145980431
```