```
import numpy as np

import pandas as pd
from scipy.optimize import minimize

# Define asset data
assets = ['Cash', 'Nifty50', 'Fixed Deposit', 'Bonds',
'Stocks', 'Gold']
expected_returns = [0.03, 0.22, 0.05, 0.057, 0.2, 0.13]  #
Replace with your expected returns
volatility = [0.01, 0.18, 0.02, 0.03, 0.2, 0.15]  # Replace
with your volatility (risk) values
liquidity = [0.9, 0.8, 0.7, 0.6, 0.5, 0.4]  # Replace with
your liquidity scores (1 = most liquid, 0 = least liquid)

# Define risk-free rate (typically the yield of a risk-free
government bond)
risk_free_rate = 0.035

# Define the portfolio optimization function
def objective(weights):
    portfolio_return = np.sum(weights * expected_returns)
    portfolio_risk = np.sqrt(np.dot(weights,
np.dot(np.diag(volatility), weights)))
    return -((portfolio_return - risk_free_rate) /
portfolio_risk)  # Maximize the return-risk ratio

# Define constraints
constraints = (
    {'type': 'eq', 'fun': lambda weights: np.sum(weights) -
1},
    {'type': 'ineq', 'fun': lambda weights: np.sum(weights *
liquidity) - 0.6},  # Adjust liquidity threshold as needed
    {'type': 'eq', 'fun': lambda weights: weights[0] - 0.0626}
)
```

```python
# Define initial guess for weights (equally weighted)
initial_weights = np.array([1.0 / len(assets) for _ in
range(len(assets))])

# Perform portfolio optimization
result = minimize(objective, initial_weights, method='SLSQP',
constraints=constraints)

# Extract the optimized portfolio weights
optimized_weights = result.x

# Print the optimized portfolio weights as percentages
portfolio_allocation = {assets[i]: optimized_weights[i] * 100
for i in range(len(assets))}
print("Optimal Portfolio Allocation:")
for asset, weight in portfolio_allocation.items():
    print(f"{asset}: {weight:.2f}%")
```