The Cholesky Decomposition for SPD Matrices

1. Find SciPy routines to perform the Cholesky factorization and solution steps separately. Non-obligatory: Implement them on your own (algorithms can be found on Wikipedia).
2. Compare the performance of the Cholesky routines to the LU routines from SciPy by testing them on large symmetric and positive definite matrices $A \in \mathbb{R}^{n\times n}$ (e.g., $A = B^TB + \delta I$ for some random $B \in \mathbb{R}^{n\times n}$ and $\delta > 0$) and some (random) right-hand side $b \in \mathbb{R}^n$.

Solution:

```
import numpy as np
import scipy.linalg as linalg
from time import time

def Aspd(n,delta):
    B = np.random.rand(n,n)
    return B.T@B + delta * np.eye(n)

def FacSol(A,b, method = "lu"):
    """
    ...
    """
    # choose correct SciPy routines
    if method == "lu":
        factor = lambda A : linalg.lu_factor(A)
        solve = lambda tup, b : linalg.lu_solve(tup, b, overwrite_b=True)
    else:
        factor = lambda A : linalg.cho_factor(A)
        solve = lambda tup, b : linalg.cho_solve(tup, b, overwrite_b=True)

    # factor
    tfac = time()
    tup = factor(A)
    tfac = time()-tfac

    # solve
    tsolve = time()
    x = solve(tup, b)
    tsolve = time()-tsolve

    return {"x": x, method: tup, "t": (tfac, tsolve)}

if __name__ == "__main__":
    n = 1000
    delta = 0.5
    runs = 10
    for i in range(runs):
        print("run", i)
        A = Aspd(n, delta)
        b = np.random.rand(n)
        print("dim, method, [t_factor t_solve], Ax==b")
        for method in ["lu", "cho"]:
            data = FacSol(A.copy(),b.copy(), method)
            print(n, method, "\t ",np.round(data["t"],4),"\t   ",np.allclose(A.dot(data["x"]),
    b))
        print("---------------------------------")
```