Python Template for Deterministic Dynamic Programming

This template assumes that the states are nonnegative whole numbers, and stages are numbered starting at 1.

```
import numpy

hugeNumber = float("inf")

Initialize all needed parameters and data
stages = number of stages

f = numpy.zeros([stages + 2, (highest-numbered state) + 1])
x = numpy.zeros([stages + 1, (highest-numbered state) + 1])

If not zero, set each f[stages+1,i] to the terminal value of being in state i at the end
For states that are not allowed, use hugenumber for minimization, -hugenumber for maximization

for t in range(stages,0,-1) :

    If necessary, determine which states are possible at stage t

    for i in (states that are possible at stage t) :

        Determine set of decisions d which are possible from this state and stage

        value = -hugeNumber if maximizing or hugenumber if minimizing

        for d in (set of allowed decisions d) :
            j = (resulting next state)
            Compute immediate costs and/or rewards from decision d
            moveValue = (immediate costs and/or rewards) + f[t+1,j]
            if moveValue > value :    (use < instead of > if minimizing)
                value = moveValue
                bestMove = d
        # End of d loop

        f[t,i] = value
        x[t,i] = bestMove

    # End of i loop

# End of t loop

print("Optimal solution is " + str(f[1,(initial state)]))
print("(something explanatory about the solution)")
i = (initial state)
for t in range(1,stages+1) :
    print(str(x[t,i]) + (some explanation))
    i = (compute next state based on decision x[t,i] being taken)
If desired, can print something about ending state here
```