Python Template for Stochastic Dynamic Programming

This template assumes that the states are nonnegative whole numbers, and stages are numbered starting at 1. Note that that the printout at the end could be may more explanatory (and less generic) for any particular problem class:

```
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) :

    for i in (possible states) :

        Determine set of decisions d which are possible in this stage/state combination
        value = -hugeNumber if maximizing or hugenumber if minimizing

        for d in (set of allowed decisions d) :

            moveValue = (net rewards/costs that are not random)

            for r in (set of random outcomes r) :
                j = (resulting next state)
                moveValue += (probability of r)*((rewards/costs depending on r) + f[t+1,j])
                # If net present value is involved, beta*f[t+1,j]) instead, where
                # beta = 1/(1 + r) is the discount factor

            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 value is " + str(f[1,(initial state)]))
print("In stage 1, decision is " + str(x[1, (initial state)])
for t in range(2,stages+1) :
    print("In stage " + str(t) + ":")
    for i in (possible states) :
      print("    If in state " + str(i) + ", decision is " + str(x[t,i])
```