File - inventory-prob-dp-npv.py

```python
import numpy

hugeNumber = float("inf")
unused     = -1000

stages             = 10
startInventory     = 1
inventoryCapacity  = 10
productionCapacity = 10

setupCost         = 10.0
variableCost      = 2.0
directHoldingCost = 0.0
salvageValue      = 2.0

minDemand = 0
maxDemand = 4

discountRate = 0.1

demandProb = numpy.array([0.30, 0.30, 0.15, 0.10, 0.05])

# End of input data section

beta = 1.0/(1.0 + discountRate)

f = numpy.zeros([stages + 2, inventoryCapacity + 1])
x = numpy.zeros([stages + 1, inventoryCapacity + 1], dtype=int)

# Set the value of ending up in each final state (not zero in this case)
for i in range(inventoryCapacity + 1):
    f[stages + 1, i] = -salvageValue * i  # Negative "cost" = benefit for each leftover unit

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

    for i in range(inventoryCapacity + 1):

        minProduction = max(0, maxDemand - i)
        maxProduction = min(productionCapacity, inventoryCapacity - i + minDemand)

        value = hugeNumber

        for p in range(minProduction, maxProduction + 1):

            # Compute production cost
            productionCost = variableCost * p
            if p > 0:
                productionCost += setupCost

            moveValue = productionCost
            # Accumulate expected present value of this decision
            for d in range(minDemand, maxDemand + 1):
                j = i + p - d
                moveValue += demandProb[d]*(directHoldingCost*j + beta*f[t + 1, j])

            if moveValue < value:
                value = moveValue
                bestMove = p

        # End of p loop

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

        # End of i loop

# End of t loop

```

File - inventory-prob-dp-npv.py

```
69 print("Optimal expected cost is " + str(f[1, startInventory]))
70 print("Period 1: produce " + str(x[1, startInventory]))
71 for t in range(2, stages + 1):
72     print("Period " + str(t) + ":")
73     sumOfxt = sum(x[t,:])
74     for i in range(inventoryCapacity + 1):
75         print("    If inventory=" + str(i) + ", produce " + str(x[t,i]))
76         sumOfxt -= x[t,i]
77         if sumOfxt <= 0 :
78             print("    If inventory>"+ str(i) + ", produce 0")
79             break
```