```
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]: def mktg_opt(u):
   ...: # Marketing Optimization in Python
   ...: # Bob Agnew, raagnew1@gmail.com, raagnew.com
   ...: from time import time
   ...: import numpy as np
   ...: import pandas as pd
   ...: print()
   ...: print()
   ...: print('Scenario # '+str(u[0]))
   ...: start_time = time()
   ...: np.random.seed(seed=2) # Set seed for repeatable results
   ...: # Risk scores range 300-850. Higher is less risky.
   ...: # These are Beta distribution simulations, not actual scores.
   ...: # Ref: fico.com/blogs/average-u-s-fico-score-ticks-706
   ...: n = u[1] # Pre-Trim Number of Prospects
   ...: prospect0 = 1 + np.arange(n)
   ...: risk0 = np.round(300+550*np.random.beta(2.57956,.91493,size=n))
   ...: # Trim all prospects with risk scores below 580
   ...: prospect = prospect0[risk0 >= 580]
   ...: risk = risk0[risk0 >= 580]
   ...: x1 = {'Pre-Trim':pd.Series(risk0),
   ...: 'Post-Trim':pd.Series(risk)}
   ...: pd.options.display.float_format = '{:,.1f}'.format
   ...: df1 = pd.DataFrame(data=x1)
   ...: print()
   ...: print()
   ...: print(" Risk Score Distribution")
   ...: print (df1.describe())
   ...:
   ...: # Probability response scores for three different credit card offers.
   ...: # These are simulations based on simple cubic risk score interpolations.
   ...: # Riskier prospects are more likely to respond.
   ...: # For given risk score, response is aligned to offer cost.
   ...: # Actual offer response scores would be modeled on various prospect
attributes.
   ...: prob1 = .001 + (.3 - .001)*((850 - risk)/550)**3
   ...: prob2 = .0001 + (.2 - .0001)*((850 - risk)/550)**3
   ...: prob3 = .00001 + (.1 - .00001)*((850 - risk)/550)**3
   ...: # Offer Unit Costs
   ...: cost = [.50,.25,.10]
   ...: # Budget Dollar Upper Bound
   ...: budget = float(u[2])
   ...: # Average Risk Score Lower Bound
   ...: avg_risk = float(u[3])
   ...: x2 = {'Measure':['Offer Dollar Budget','Average Risk Score'],
   ...: 'Bound':[budget,avg_risk]}
   ...: df2 = pd.DataFrame(data=x2)
   ...: print()
   ...: print()
   ...: print(' Stipulated Constraints')
```

```
   ...: print(df2.to_string(index=False,justify='right'))
   ...:
   ...: # Dual Optimization
   ...: n = np.size(prospect)
   ...: z = np.zeros(n)
   ...: v = np.full(n,avg_risk)
   ...: def dual(y):
   ...: d1 = prob1 - cost[0]*y[0] - prob1*(v - risk)*y[1]
   ...: d2 = prob2 - cost[1]*y[0] - prob2*(v - risk)*y[1]
   ...: d3 = prob3 - cost[2]*y[0] - prob3*(v - risk)*y[1]
   ...: d = np.array([z,d1,d2,d3])
   ...: val = budget*y[0] + sum(np.amax(d,axis=0))
   ...: return val
   ...: from scipy.optimize import minimize
   ...: bnds = ((0,None),(0,None))
   ...: res = minimize(dual,(0,0),method='L-BFGS-B',bounds=bnds)
   ...: print()
   ...: print()
   ...: print(' Quasi-Optimal Dual Solution')
   ...: txt = 'Minimum Dual Value = {xxx:,.1f}'+' (Compare to Total Expected
Responses)'
   ...: print(txt.format(xxx = res.fun))
   ...: print('Minimum Dual Parameters = '+str(res.x))
   ...: y = res.x
   ...:
   ...: # Primal Assignments
   ...: d1 = prob1 - cost[0]*y[0] - prob1*(v - risk)*y[1]
   ...: d2 = prob2 - cost[1]*y[0] - prob2*(v - risk)*y[1]
   ...: d3 = prob3 - cost[2]*y[0] - prob3*(v - risk)*y[1]
   ...: d = np.array([d1,d2,d3])
   ...: w = np.amax(d,axis=0)
   ...: offer = 1 + np.argmax(d,axis=0)
   ...: order = np.flip(np.argsort(w)) # Offers in descending order
   ...: w = w[order]
   ...: prospect = prospect[order]
   ...: risk = risk[order]
   ...: offer = offer[order]
   ...: prob1 = prob1[order]
   ...: prob2 = prob2[order]
   ...: prob3 = prob3[order]
   ...: c = cost[0]*(offer==1) + cost[1]*(offer==2) + cost[2]*(offer==3)
   ...: cumcost = np.cumsum(c)
   ...: offer[np.logical_or(w < 0,cumcost > budget)] = 0
   ...: offers1 = float(sum(offer==1))
   ...: offers2 = float(sum(offer==2))
   ...: offers3 = float(sum(offer==3))
   ...: total_offers = offers1 + offers2 + offers3
   ...: resp1 = sum(prob1[offer==1])
   ...: resp2 = sum(prob2[offer==2])
   ...: resp3 = sum(prob3[offer==3])
   ...: total_resp = resp1 + resp2 + resp3
   ...: cost1 = cost[0]*offers1
   ...: cost2 = cost[1]*offers2
   ...: cost3 = cost[2]*offers3
   ...: total_cost = cost1 + cost2 + cost3
```

```
   ...: risk1 = sum(prob1[offer==1]*risk[offer==1])/(resp1 + 1e-15)
   ...: risk2 = sum(prob2[offer==2]*risk[offer==2])/(resp2 + 1e-15)
   ...: risk3 = sum(prob3[offer==3]*risk[offer==3])/(resp3 + 1e-15)
   ...: average_risk = (resp1*risk1 + resp2*risk2 + resp3*risk3)/total_resp
   ...: x3 = {'Offer':['# 1','# 2','# 3','Total'],
   ...: 'Quantity':[offers1,offers2,offers3,total_offers],
   ...: 'Total $ Cost':[cost1,cost2,cost3,total_cost],
   ...: 'Expected Responses':[resp1,resp2,resp3,total_resp],
   ...: 'Avg Responder Risk Score':[risk1,risk2,risk3,average_risk]}
   ...: df3 = pd.DataFrame(data=x3)
   ...: print()
   ...: print()
   ...: print(' Quasi-Optimal Primal Solution')
   ...: print(df3.to_string(index=False,justify='right'))
   ...: print()
   ...: print()
   ...: txt = 'Cost Per Response = ${xxx:,.1f}'
   ...: print(txt.format(xxx = total_cost/total_resp))
   ...: print()
   ...: print()
   ...: offered = np.arange(int(total_offers))
   ...: prospect = prospect[offered]
   ...: risk = risk[offered]
   ...: offer = offer[offered]
   ...: order = np.argsort(prospect,axis=0)
   ...: prospect = prospect[order]
   ...: risk = risk[order]
   ...: offer = offer[order]
   ...: x4 = {'Prospect':prospect,'Risk':risk,'Offer':offer}
   ...: df4 = pd.DataFrame(data=x4)
   ...: print('First 50 Sorted Prospect Offers')
   ...: pd.options.display.float_format = '{:,.0f}'.format
   ...: print(df4.head(50).to_string(index=False,justify='right'))
   ...: print()
   ...: print()
   ...: end_time = time()
   ...: print('Elapsed Seconds = '+str(end_time - start_time))
   ...: print()
   ...: print()
   ...: # To save entire campaign list:
   ...: # import csv
   ...: # df4.to_csv('c:/Marketing Optimization/Campaign_List.csv')
   ...: return
   ...:
   ...:

In [2]: mktg_opt([1,1000000,50000,700])


Scenario # 1


Risk Score Distribution
Pre-Trim Post-Trim
count 1,000,000.0 843,539.0
```

mean 706.0 743.7
std 114.0 75.3
min 304.0 580.0
25% 634.0 686.0
50% 732.0 756.0
75% 801.0 809.0
max 850.0 850.0

Stipulated Constraints
Measure Bound
Offer Dollar Budget 50,000.0
Average Risk Score 700.0

Quasi-Optimal Dual Solution
Minimum Dual Value = 760.4 (Compare to Total Expected Responses)
Minimum Dual Parameters = [0.01224113 0.01934262]

Quasi-Optimal Primal Solution
Offer Quantity Total $ Cost Expected Responses Avg Responder Risk Score
1 23,900.0 11,950.0 115.1 721.4
2 112,140.0 28,035.0 474.6 697.1
3 100,150.0 10,015.0 168.4 694.3
Total 236,190.0 50,000.0 758.2 700.1

Cost Per Response = $65.9

First 50 Sorted Prospect Offers
Prospect Risk Offer
1 724 1
25 742 3
26 739 3
27 743 3
28 677 3
31 743 3
32 699 2
40 738 3
41 747 3
45 666 3
53 669 3
54 733 3
56 679 2
59 725 1
63 691 2
68 704 2
69 666 3
78 689 2
85 707 2
87 708 2
97 670 3
98 722 1

```
100 718 1
104 708 2
109 712 2
110 676 3
112 728 3
113 731 3
114 719 1
119 703 2
120 739 3
129 666 3
132 719 1
138 679 2
142 734 3
144 700 2
153 743 3
155 670 3
156 747 3
161 728 3
162 733 3
163 696 2
169 727 2
176 742 3
190 748 3
192 740 3
193 734 3
194 721 1
198 720 1
202 694 2

Elapsed Seconds = 63.48202323913574
```

```
In [3]: mktg_opt([2,1000000,250000,700])
```

```
Scenario # 2

Risk Score Distribution
Pre-Trim Post-Trim
count 1,000,000.0 843,539.0
mean 706.0 743.7
std 114.0 75.3
min 304.0 580.0
25% 634.0 686.0
50% 732.0 756.0
75% 801.0 809.0
max 850.0 850.0

Stipulated Constraints
Measure Bound
Offer Dollar Budget 250,000.0
```

Average Risk Score 700.0

Quasi-Optimal Dual Solution
Minimum Dual Value = 2,539.5 (Compare to Total Expected Responses)
Minimum Dual Parameters = [0.0062182 0.01473683]

Quasi-Optimal Primal Solution

Offer	Quantity	Total $ Cost	Expected Responses	Avg Responder Risk Score
# 1	490,981.0	245,490.5	2,346.7	704.5
# 2	14,753.0	3,688.2	149.8	646.9
# 3	8,211.0	821.1	44.8	641.5
Total	513,945.0	249,999.9	2,541.3	700.0

Cost Per Response = $98.4

First 50 Sorted Prospect Offers

Prospect	Risk	Offer
1	724	1
4	781	1
8	662	1
9	749	1
11	775	1
12	775	1
14	761	1
17	778	1
18	640	3
19	843	1
20	774	1
23	647	2
25	742	1
26	739	1
27	743	1
28	677	1
29	641	3
30	844	1
31	743	1
32	699	1
33	762	1
34	663	1
35	847	1
38	653	1
40	738	1
41	747	1
42	757	1
43	654	1
44	849	1
45	666	1
46	640	3
47	850	1
48	655	1
52	774	1

```
53 669 1
54 733 1
56 679 1
59 725 1
63 691 1
64 792 1
68 704 1
69 666 1
75 849 1
78 689 1
80 750 1
84 664 1
85 707 1
87 708 1
88 755 1
90 791 1
```

Elapsed Seconds = 33.966370820999146

```
In [4]: mktg_opt([3,1000000,500000,700])
```

Scenario # 3

Risk Score Distribution

	Pre-Trim	Post-Trim
count	1,000,000.0	843,539.0
mean	706.0	743.7
std	114.0	75.3
min	304.0	580.0
25%	634.0	686.0
50%	732.0	756.0
75%	801.0	809.0
max	850.0	850.0

Stipulated Constraints

Measure	Bound
Offer Dollar Budget	500,000.0
Average Risk Score	700.0

Quasi-Optimal Dual Solution
Minimum Dual Value = 3,254.0 (Compare to Total Expected Responses)
Minimum Dual Parameters = [0. 0.01470588]

Quasi-Optimal Primal Solution

Offer	Quantity	Total $ Cost	Expected Responses	Avg Responder Risk Score
# 1	754,600.0	377,300.0	3,269.3	699.7
# 2	0.0	0.0	0.0	0.0

# 3	0.0	0.0	0.0	0.0
Total	754,600.0	377,300.0	3,269.3	699.7

Cost Per Response = $115.4

First 50 Sorted Prospect Offers

Prospect	Risk	Offer
1	724	1
2	807	1
4	781	1
5	835	1
6	797	1
7	833	1
8	662	1
9	749	1
10	840	1
11	775	1
12	775	1
13	839	1
14	761	1
15	824	1
16	810	1
17	778	1
18	640	1
19	843	1
20	774	1
21	836	1
23	647	1
25	742	1
26	739	1
27	743	1
28	677	1
29	641	1
30	844	1
31	743	1
32	699	1
33	762	1
34	663	1
35	847	1
36	830	1
37	807	1
38	653	1
39	818	1
40	738	1
41	747	1
42	757	1
43	654	1
44	849	1
45	666	1
46	640	1
47	850	1
48	655	1
50	634	1

52 774 1
53 669 1
54 733 1
56 679 1

Elapsed Seconds = 46.979647397994995

In [5]: mktg_opt([4,1000000,50000,675])

Scenario # 4

Risk Score Distribution
Pre-Trim Post-Trim
count 1,000,000.0 843,539.0
mean 706.0 743.7
std 114.0 75.3
min 304.0 580.0
25% 634.0 686.0
50% 732.0 756.0
75% 801.0 809.0
max 850.0 850.0

Stipulated Constraints
Measure Bound
Offer Dollar Budget 50,000.0
Average Risk Score 675.0

Quasi-Optimal Dual Solution
Minimum Dual Value = 1,203.6 (Compare to Total Expected Responses)
Minimum Dual Parameters = [0.01816016 0.01708912]

Quasi-Optimal Primal Solution
Offer Quantity Total $ Cost Expected Responses Avg Responder Risk Score
1 0.0 0.0 0.0 0.0
2 157,152.0 39,288.0 948.3 676.0
3 107,120.0 10,712.0 262.6 669.8
Total 264,272.0 50,000.0 1,210.9 674.6

Cost Per Response = $41.3

First 50 Sorted Prospect Offers
Prospect Risk Offer
1 724 3
8 662 2
18 640 3

```
23 647 3
28 677 2
29 641 3
32 699 2
34 663 2
38 653 2
43 654 2
45 666 2
46 640 3
48 655 2
50 634 3
53 669 2
54 733 3
56 679 2
59 725 3
63 691 2
68 704 2
69 666 2
78 689 2
84 664 2
85 707 2
87 708 2
97 670 2
98 722 3
100 718 3
104 708 2
107 647 3
109 712 3
110 676 2
112 728 3
113 731 3
114 719 3
119 703 2
126 655 2
129 666 2
131 662 2
132 719 3
138 679 2
140 646 3
141 656 2
142 734 3
144 700 2
151 651 2
155 670 2
161 728 3
162 733 3
163 696 2


Elapsed Seconds = 38.114755153656006
```

```
In [6]: mktg_opt([5,5000000,250000,700])
```

Scenario # 5

Risk Score Distribution

	Pre-Trim	Post-Trim
count	5,000,000.0	4,219,575.0
mean	706.0	743.7
std	114.0	75.3
min	301.0	580.0
25%	634.0	686.0
50%	732.0	756.0
75%	801.0	809.0
max	850.0	850.0

Stipulated Constraints

Measure	Bound
Offer Dollar Budget	250,000.0
Average Risk Score	700.0

Quasi-Optimal Dual Solution
Minimum Dual Value = 3,800.0 (Compare to Total Expected Responses)
Minimum Dual Parameters = [0.01223409 0.01929953]

Quasi-Optimal Primal Solution

Offer	Quantity	Total $ Cost	Expected Responses	Avg Responder Risk Score
# 1	119,704.0	59,852.0	576.4	721.4
# 2	558,827.0	139,706.8	2,365.9	697.1
# 3	504,412.0	50,441.2	856.5	693.8
Total	1,182,943.0	250,000.0	3,798.9	700.0

Cost Per Response = $65.8

First 50 Sorted Prospect Offers

Prospect	Risk	Offer
1	724	1
25	742	3
26	739	3
27	743	3
28	677	3
31	743	3
32	699	2
40	738	3
41	747	3
45	666	3
53	669	3
54	733	3
56	679	2
59	725	1
63	691	2

```
68 704 2
69 666 3
78 689 2
85 707 2
87 708 2
97 670 3
98 722 1
100 718 1
104 708 2
109 712 2
110 676 3
112 728 3
113 731 3
114 719 1
119 703 2
120 739 3
129 666 3
132 719 1
138 679 2
142 734 3
144 700 2
153 743 3
155 670 3
156 747 3
161 728 3
162 733 3
163 696 2
169 727 2
176 742 3
190 748 3
192 740 3
193 734 3
194 721 1
198 720 1
202 694 2

Elapsed Seconds = 315.75010776519775
```

In [7]: