In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from mpl_toolkits.mplot3d import Axes3D
```

Problem 5.1

If the individual only live for T = 1, his value function will be:

$$\max_{c_1 \in [0, W_1]} u(c_1)$$

Since $u(.)$ is a monotonically increasing function, the maximum value of $c_1 = W_1$ would give the max utility.

The equivalent representation of the problem in terms of W_2 is:

$$\max_{W_2 \in [0, W_1]} u(W_1 - W_2)$$

The optimal solution would be when W_2 takes the smallest value = 0.

Problem 5.2

In this case also, optimal utility is achieved when all the cake is consumed by time T. Thus for optimal utility, $W_3 = 0$.
The following condition charaterizes W_2 when $T = 1$:

$$\max_{W_2 \in [0, W_1]} u(W_1 - W_2) + \beta u(W_2)$$

Problem 5.3

The condition characterizing the optimal amount of cake to leave for the next period in each of the three periods is given by:

T = 3: All cake has to be consumed for max utility, hence:

$$W_4 = 0$$

T = 2:

$$\max_{W_3 \in [0, W_2]} u(W_2 - W_3) + \beta u(W_3)$$

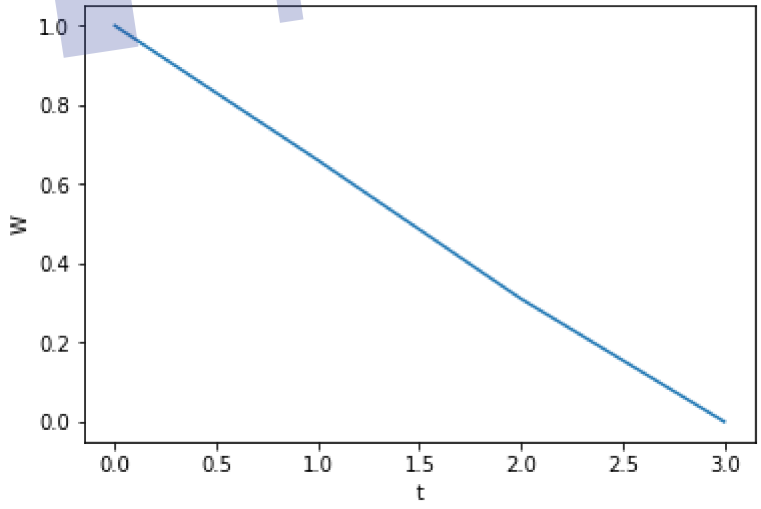
T = 1:

$$\max_{\substack{W_2 \in [0, W_1] \\ W_3 \in [0, W_2]}} u(W_1 - W_2) + \beta u(W_2 - W_3) + \beta^2 u(W_3)$$

Differentiating the equation from $T = 1$ w.r.t W_2 and W_3 and setting it to zero. Then substituting $u(c) = ln(c)$ and the values $W_1 = 1, \beta = 0.9$we get $W_1 = 1, W_2 = 0.66, W_3 = 0.31, W_4 = 0$correspondingly, $c_1 = 0.34, c_2 = 0.35, c_3 = 0.31$

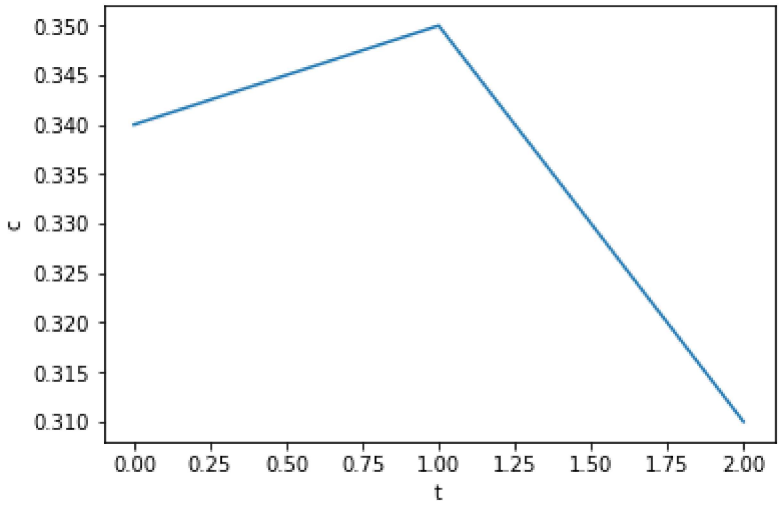
In [2]:
```
W = [1, 0.66, 0.31, 0]
c = 0.34, 0.35, 0.31
T = [0, 1, 2, 3]
plt.plot(T, W)
plt.xlabel("t")
plt.ylabel("W")
```

Out[2]: Text(0, 0.5, 'W')



In [3]:
```
plt.plot(T[:3], c)
plt.xlabel("t")
plt.ylabel("c")
```

Out[3]: Text(0, 0.5, 'c')



Problem 5.4

The condition that characterizes the optimal policy function in period $T-1$ can be obtained from the first order condition. It is given as the function $\Phi_{T-1}(W_{t-1})$ that satisfies:

$$\frac{\partial u(W_{T-1} - \Phi_{T-1}(W_{t-1}))}{\partial W_{T-1}} + \beta \frac{u(\Phi_{T-1}(W_{t-1}))}{\partial W_{T-1}} = 0$$

The value of V_{T-1} in terms of $\Phi_{T-1}(W_{t-1})$ is given by (note that W_T is already optimized by taking the Φ function, so we may omit the maximization part of the equation):

$$V_{T-1} = u(W_{T-1} - \Phi_{T-1}(W_{t-1})) + \beta u(\Phi_{T-1}(W_{T-1}))$$

Problem 5.5

$$\Phi_T(\bar{W}) = 0$$
$$\Phi_{T-1}(\bar{W}) = \frac{\beta}{1+\beta}\bar{W}$$

So clearly $\Phi_T(\bar{W}) \neq \Phi_{T-1}(\bar{W})$. Also,

$$V_T(\bar{W}) = u(\bar{W})$$
$$V_{T-1}(\bar{W}) = u(\bar{W} - \Phi_{T-1}(\bar{W})) + \beta u(\Phi_{T-1}(\bar{W}))$$

So clearly $V_T(\bar{W}) \neq V_{T-1}(\bar{W})$

Problem 5.6

The finite horizon Bellman equation is:

$$V_{T-2}(W_{T-2}) = \max_{\substack{W_T \in [0, W_{T-1}] \\ W_{T-1} \in [0, W_{T-2}]}} ln(W_{T-2} - W_{T-1}) + \beta ln(W_{T-1} - W_T) + \beta^2 ln(W_T)$$

Substituting for the optimal values of $W_T = \Phi_{T-1}(W_{T-1})$ from the previous question we get:

$$V_{T-2}(W_{T-2}) = \max_{W_{T-1} \in [0, W_{T-2}]} ln(W_{T-2} - W_{T-1}) + \beta ln(W_{T-1} - \frac{\beta}{1+\beta} W_{T-1}) + \beta^2 ln(\frac{\beta}{1+\beta} W_{T-1})$$

We can obtain the optimal value of W_{T-1} by differentiating the above equation and setting the differential to zero to meet the first order condition. Then we get $\Phi_{T-2}(W_{T-2})$ as:

$$W_{T-1} = \Phi_{T-2}(W_{T-2}) = \frac{\beta + \beta^2}{1 + \beta + \beta^2} W_{T-2}$$

Pluggging this in, we get the analytical solution for V_{T-2} :

$$V_{T-2}(W_{T-2}) = ln(\frac{W_{T-2}}{1 + \beta + \beta^2}) + \beta ln(\frac{\beta W_{T-2}}{1 + \beta + \beta^2}) + \beta^2 ln(\frac{\beta^2 W_{T-2}}{1 + \beta + \beta^2})$$

Problem 5.7

The analytical solutions for $\Phi_{T-s}(W_{T-s})$ and $V_{T-s}(W_{T-s})$ obtained through induction are:

$$\Phi_{T-s}(W_{T-s}) = \frac{\sum_{i=1}^{s} \beta^i}{\sum_{i=0}^{s} \beta^i} W_{T-s}$$

$$V_{T-s}(W_{T-s}) = \sum_{i=0}^{s} \beta^i ln\left\{ \frac{\beta^i W_{T-s}}{\sum_{k=0}^{s} \beta^k} \right\}$$

When the horizon tends to infinity we get functions that do not vary with time but only with W_{T-s}:

$$\Phi(W_{T-s}) = \beta W_{T-s}$$

$$V(W_{T-s}) = \left(\frac{1}{1-\beta} \right) ln((1-\beta) W_{T-s}) + \left(\frac{\beta}{(1-\beta)^2} \right) ln(\beta)$$

Problem 5.8

Bellman equation for the infinite horizon is given as:

$$V(W) = \max_{W' \in [0, W]} u(W - W') + \beta V(W')$$

Problem 5.9 to 5.15

In [4]:
```
#Problem 5.9
w_min, w_max = 0.01, 1
N = 100
beta = 0.9
w_vec = np.linspace(0.01, 1, 100)
print(w_vec)
```

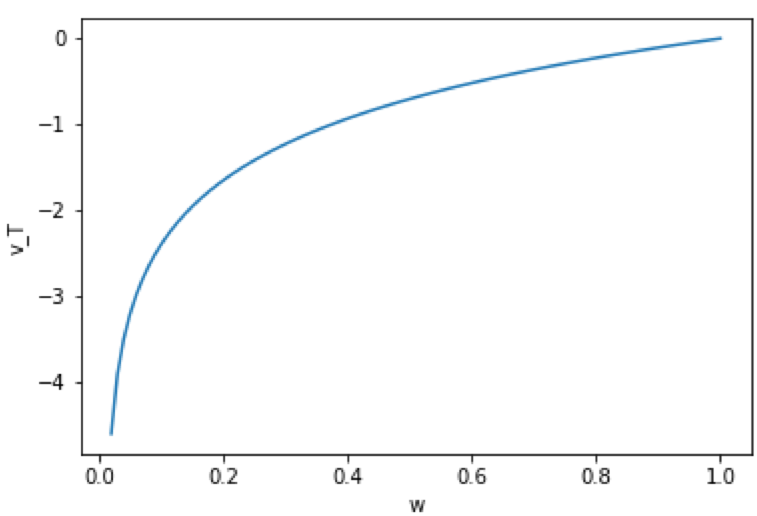
```
[0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1  0.11 0.12 0.13 0.14
 0.15 0.16 0.17 0.18 0.19 0.2  0.21 0.22 0.23 0.24 0.25 0.26 0.27 0.28
 0.29 0.3  0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.4  0.41 0.42
 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.5  0.51 0.52 0.53 0.54 0.55 0.56
 0.57 0.58 0.59 0.6  0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.7
 0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.8  0.81 0.82 0.83 0.84
 0.85 0.86 0.87 0.88 0.89 0.9  0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98
 0.99 1.  ]
```

In [5]:

```
#Problem 5.10
def log_utility(c):
    return np.log(c)
N = 100
beta = 0.9

#calculating the utility of consumption matrix
c_mat = np.tile(w_vec.reshape(N,1), (1,N)) - np.tile(w_vec.reshape(1,N), (N,1
))
c_pos = c_mat <= 0
c_mat[c_pos] = 1e-10
u_mat = log_utility(c_mat)

#v_(t+1) is a zero matrix
v_init = np.zeros((N,))
v_prime = np.tile(v_init.reshape((1,N)), (N,1))
v_new = (u_mat + beta * v_prime).max(axis=1)
max_ind = np.argmax(u_mat + beta * v_prime, axis=1)
v_new_510_store = v_new.copy()

#policy function and value function:
print(w_vec[max_ind])
print(v_new)
plt.plot(w_vec[1:], v_new[1:])
plt.xlabel('w')
plt.ylabel('v_T')
```

```
[0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01]
[-2.30258509e+01 -4.60517019e+00 -3.91202301e+00 -3.50655790e+00
 -3.21887582e+00 -2.99573227e+00 -2.81341072e+00 -2.65926004e+00
 -2.52572864e+00 -2.40794561e+00 -2.30258509e+00 -2.20727491e+00
 -2.12026354e+00 -2.04022083e+00 -1.96611286e+00 -1.89711998e+00
 -1.83258146e+00 -1.77195684e+00 -1.71479843e+00 -1.66073121e+00
 -1.60943791e+00 -1.56064775e+00 -1.51412773e+00 -1.46967597e+00
 -1.42711636e+00 -1.38629436e+00 -1.34707365e+00 -1.30933332e+00
 -1.27296568e+00 -1.23787436e+00 -1.20397280e+00 -1.17118298e+00
 -1.13943428e+00 -1.10866262e+00 -1.07880966e+00 -1.04982212e+00
 -1.02165125e+00 -9.94252273e-01 -9.67584026e-01 -9.41608540e-01
 -9.16290732e-01 -8.91598119e-01 -8.67500568e-01 -8.43970070e-01
 -8.20980552e-01 -7.98507696e-01 -7.76528789e-01 -7.55022584e-01
 -7.33969175e-01 -7.13349888e-01 -6.93147181e-01 -6.73344553e-01
 -6.53926467e-01 -6.34878272e-01 -6.16186139e-01 -5.97837001e-01
 -5.79818495e-01 -5.62118918e-01 -5.44727175e-01 -5.27632742e-01
 -5.10825624e-01 -4.94296322e-01 -4.78035801e-01 -4.62035460e-01
 -4.46287103e-01 -4.30782916e-01 -4.15515444e-01 -4.00477567e-01
 -3.85662481e-01 -3.71063681e-01 -3.56674944e-01 -3.42490309e-01
 -3.28504067e-01 -3.14710745e-01 -3.01105093e-01 -2.87682072e-01
 -2.74436846e-01 -2.61364764e-01 -2.48461359e-01 -2.35722334e-01
 -2.23143551e-01 -2.10721031e-01 -1.98450939e-01 -1.86329578e-01
 -1.74353387e-01 -1.62518929e-01 -1.50822890e-01 -1.39262067e-01
 -1.27833372e-01 -1.16533816e-01 -1.05360516e-01 -9.43106795e-02
 -8.33816089e-02 -7.25706928e-02 -6.18754037e-02 -5.12932944e-02
 -4.08219945e-02 -3.04592075e-02 -2.02027073e-02 -1.00503359e-02]
```


Out[5]: Text(0, 0.5, 'v_T')



In [6]:
```
#Problem 5.11
dist_1 = sum((v_new - v_init) ** 2)
print(dist_1)
```

```
709.115921707568
```

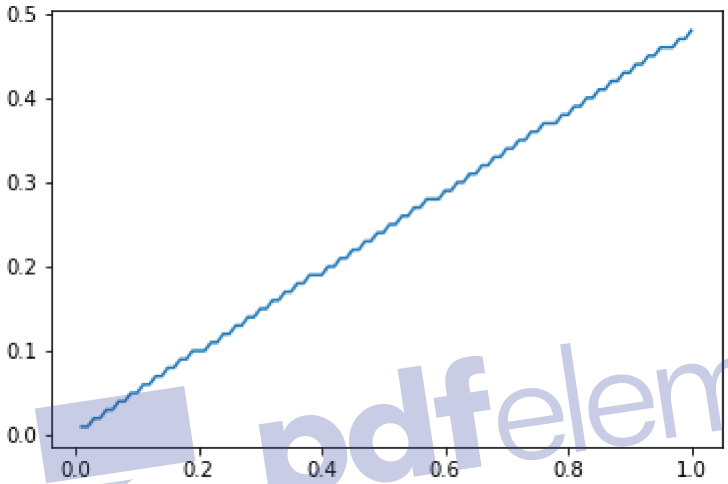
In [7]:
```
#Problem 5.12
#using v_new from previous question as v_init in this question:
v_init = v_new.copy()
v_prime = np.tile(v_init.reshape((1,N)), (N,1))
v_prime[c_pos] = -9e+4
v_new = (u_mat + beta * v_prime).max(axis=1)
plt.plot(w_vec, v_new)
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x1b9b0e898d0>]
```

In [8]:
```
max_ind = np.argmax((u_mat + beta * v_prime), axis=1)
w_dash_vec = w_vec[max_ind]
print(w_dash_vec)
plt.plot(w_vec, w_dash_vec)
```

```
[0.01 0.01 0.02 0.02 0.03 0.03 0.04 0.04 0.05 0.05 0.06 0.06 0.07 0.07
 0.08 0.08 0.09 0.09 0.1  0.1  0.1  0.11 0.11 0.12 0.12 0.13 0.13 0.14
 0.14 0.15 0.15 0.16 0.16 0.17 0.17 0.18 0.18 0.19 0.19 0.19 0.2  0.2
 0.21 0.21 0.22 0.22 0.23 0.23 0.24 0.24 0.25 0.25 0.26 0.26 0.27 0.27
 0.28 0.28 0.28 0.29 0.29 0.3  0.3  0.31 0.31 0.32 0.32 0.33 0.33 0.34
 0.34 0.35 0.35 0.36 0.36 0.37 0.37 0.37 0.38 0.38 0.39 0.39 0.4  0.4
 0.41 0.41 0.42 0.42 0.43 0.43 0.44 0.44 0.45 0.45 0.46 0.46 0.46 0.47
 0.47 0.48]
```

Out[8]: [<matplotlib.lines.Line2D at 0x1b9b0ef2048>]



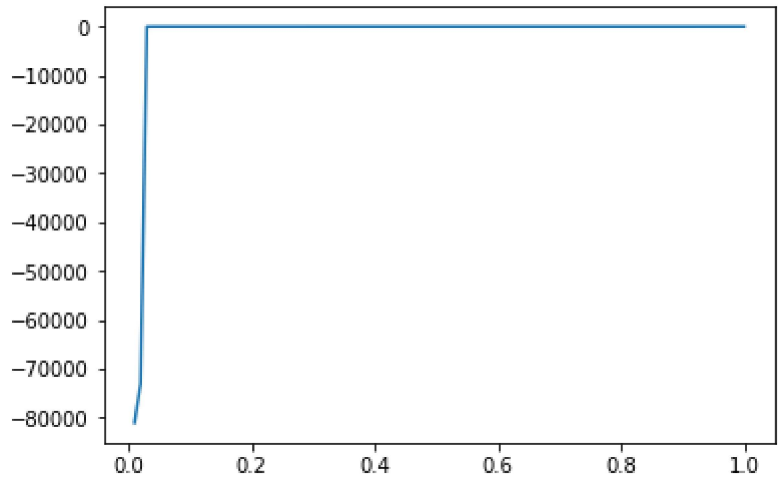
In [9]:
```
dist_2 = sum((v_new - v_init) ** 2)
print(dist_2, dist_1)
```

```
6561000945.781889 709.115921707568
```

δ_{t-1} (dist2) *is larger then* $\delta(t)(dist_1), probablybecauseweassumedthat$V(W_{T+1}) = 0$ but going forward we can expect the distance value to decrease!

In [10]:
```
#Problem 5.13
#using v_new from previous question as v_init in this question:
v_init = v_new.copy()
v_prime = np.tile(v_init.reshape((1,N)), (N,1))
v_prime[c_pos] = -9e+4
v_new = (u_mat + beta * v_prime).max(axis=1)
plt.plot(w_vec,v_new)
```

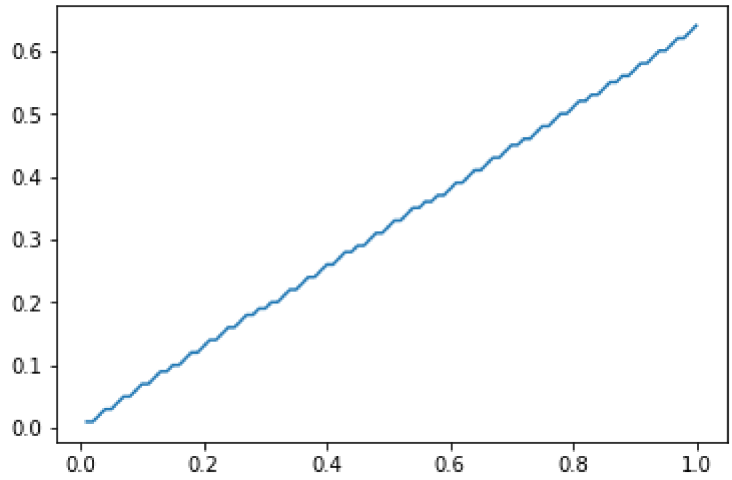
Out[10]: [<matplotlib.lines.Line2D at 0x1b9b0f4c1d0>]



In [11]:
```
ind = np.argmax((u_mat + beta * v_prime), axis=1)
w_dash_vec = w_vec[ind]
print(w_dash_vec)
plt.plot(w_vec, w_dash_vec)
```

```
[0.01 0.01 0.02 0.03 0.03 0.04 0.05 0.05 0.06 0.07 0.07 0.08 0.09 0.09
 0.1  0.1  0.11 0.12 0.12 0.13 0.14 0.14 0.15 0.16 0.16 0.17 0.18 0.18
 0.19 0.19 0.2  0.2  0.21 0.22 0.22 0.23 0.24 0.24 0.25 0.26 0.26 0.27
 0.28 0.28 0.29 0.29 0.3  0.31 0.31 0.32 0.33 0.33 0.34 0.35 0.35 0.36
 0.36 0.37 0.37 0.38 0.39 0.39 0.4  0.41 0.41 0.42 0.43 0.43 0.44 0.45
 0.45 0.46 0.46 0.47 0.48 0.48 0.49 0.5  0.5  0.51 0.52 0.52 0.53 0.53
 0.54 0.55 0.55 0.56 0.56 0.57 0.58 0.58 0.59 0.6  0.6  0.61 0.62 0.62
 0.63 0.64]
```

Out[11]: [<matplotlib.lines.Line2D at 0x1b9b0fa9940>]



In [12]:
```
dist_3 = sum((v_new - v_init) ** 2)
print(dist_3, dist_2, dist_1)
```

```
5314410944.039497 6561000945.781889 709.115921707568
```

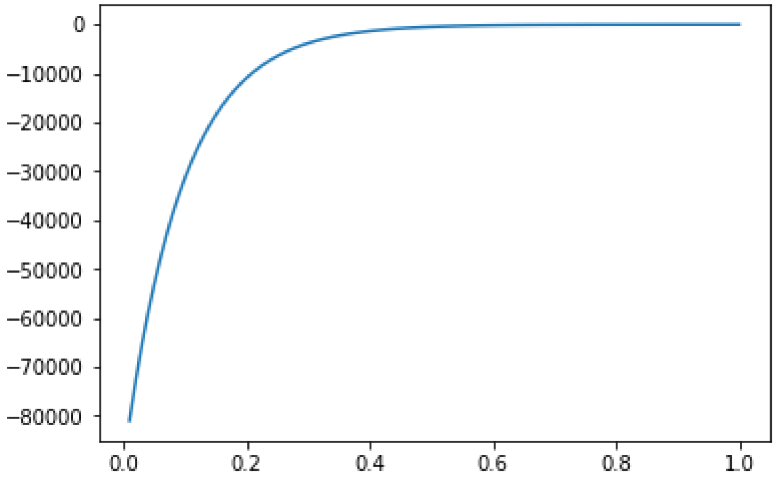
We can see that after the abberation at $t = T$, the distance function starts to decrease

In [13]:
```
#Problem 5.14
N = 100
beta = 0.9
w_vec = np.linspace(0.01, 1, 100)
c_mat = np.tile(w_vec.reshape(N,1), (1,N)) - np.tile(w_vec.reshape(1,N), (N,1
))
c_pos = c_mat <= 0
c_mat[c_pos] = 1e-10
u_mat = log_utility(c_mat)

dist = 10
maxiters = 500
toler = 1e-9
vf_iter = 0
v_init = v_new_510_store #starting from the v value obtained in 5.10 i.e from
 t=T-1

while dist > toler and vf_iter < maxiters:
    #value function iteration
    v_prime = np.tile(v_init.reshape((1,N)), (N,1))
    v_prime[c_pos] = -9e+4
    v_new = (u_mat + beta * v_prime).max(axis=1)
    max_ind = np.argmax((u_mat + beta * v_prime), axis=1)

    dist = sum((v_new - v_init) ** 2)
    v_init = v_new.copy()
    print("value : iters = {} distance = {}".format(vf_iter, dist))
    vf_iter += 1
plt.plot(w_vec, v_new)
```

```
value : iters = 0 distance = 6561000945.781889
value : iters = 1 distance = 5314410944.039497
value : iters = 2 distance = 4304672974.9382925
value : iters = 3 distance = 3486785183.7277846
value : iters = 4 distance = 2824296050.897787
value : iters = 5 distance = 2287679838.891945
value : iters = 6 distance = 1853020697.3497062
value : iters = 7 distance = 1500946785.8739464
value : iters = 8 distance = 1215766912.6195722
value : iters = 9 distance = 984771211.6942745
value : iters = 10 distance = 797664691.2687932
value : iters = 11 distance = 646108407.7856617
value : iters = 12 distance = 523347816.6077542
value : iters = 13 distance = 423911736.6686994
value : iters = 14 distance = 343368511.04474694
value : iters = 15 distance = 278128497.59140617
value : iters = 16 distance = 225284086.23610365
value : iters = 17 distance = 182480112.64772516
value : iters = 18 distance = 147808893.69213763
value : iters = 19 distance = 119725206.05318752
value : iters = 20 distance = 96977418.92497352
value : iters = 21 distance = 78551711.24183308
value : iters = 22 distance = 63626887.91543276
value : iters = 23 distance = 51537780.92607228
value : iters = 24 distance = 41745604.176571436
value : iters = 25 distance = 33813940.92880815
value : iters = 26 distance = 27389293.624303818
value : iters = 27 distance = 22185329.239664424
value : iters = 28 distance = 17970118.025825135
value : iters = 29 distance = 14555796.885991575
value : iters = 30 distance = 11790196.709755499
value : iters = 31 distance = 9550060.42021094
value : iters = 32 distance = 7735549.980065521
value : iters = 33 distance = 6265796.495826588
value : iters = 34 distance = 5075296.147908674
value : iters = 35 distance = 4110990.84274331
value : iters = 36 distance = 3329903.4932164014
value : iters = 37 distance = 2697222.716040627
value : iters = 38 distance = 2184751.2725040405
value : iters = 39 distance = 1769649.3785211793
value : iters = 40 distance = 1433416.8245609475
value : iters = 41 distance = 1161068.4446418437
value : iters = 42 distance = 940466.2379002784
value : iters = 43 distance = 761778.4413498973
value : iters = 44 distance = 617041.3134588508
value : iters = 45 distance = 499804.2307559878
value : iters = 46 distance = 404842.18572186807
value : iters = 47 distance = 327922.9248135271
value : iters = 48 distance = 265618.34109206876
value : iters = 49 distance = 215151.6253035357
value : iters = 50 distance = 174273.5786540097
value : iters = 51 distance = 141162.35456735003
value : iters = 52 distance = 114342.25641706752
value : iters = 53 distance = 92617.97099667123
value : iters = 54 distance = 75021.29423001739
value : iters = 55 distance = 60767.97968073523
value : iters = 56 distance = 49222.78934895594
```

```
value : iters = 57 distance = 39871.17931134257
value : iters = 58 distance = 32296.36970768011
value : iters = 59 distance = 26160.767817591048
value : iters = 60 distance = 21190.924815189734
value : iters = 61 distance = 17165.34533609862
value : iters = 62 distance = 13904.62032132924
value : iters = 63 distance = 11263.426904507629
value : iters = 64 distance = 9124.054443167624
value : iters = 65 distance = 7391.156076956817
value : iters = 66 distance = 5987.502409217101
value : iters = 67 distance = 4850.5354306858235
value : iters = 68 distance = 3929.5846846371846
value : iters = 69 distance = 3183.6078663379653
value : iters = 70 distance = 2579.3588032469493
value : iters = 71 distance = 2089.9100120726293
value : iters = 72 distance = 1693.447540140075
value : iters = 73 distance = 1372.3039664698406
value : iters = 74 distance = 1112.1696255540621
value : iters = 75 distance = 901.4513500842971
value : iters = 76 distance = 730.7580786507208
value : iters = 77 distance = 592.4855907819593
value : iters = 78 distance = 480.4750607439696
value : iters = 79 distance = 389.73496104760494
value : iters = 80 distance = 316.2214201794729
value : iters = 81 distance = 256.6620330193205
value : iters = 82 distance = 208.40688465940326
value : iters = 83 distance = 169.3004243303454
value : iters = 84 distance = 137.607669218765
value : iters = 85 distance = 111.92170519758028
value : iters = 86 distance = 91.09167979800623
value : iters = 87 distance = 74.198991727148
value : iters = 88 distance = 60.497628207341286
value : iters = 89 distance = 49.36373949381441
value : iters = 90 distance = 40.22719727668055
value : iters = 91 distance = 32.11388221617012
value : iters = 92 distance = 25.532077103774554
value : iters = 93 distance = 19.759463045202654
value : iters = 94 distance = 15.149394755932851
value : iters = 95 distance = 11.169967257907043
value : iters = 96 distance = 8.017010168367104
value : iters = 97 distance = 5.338935567489141
value : iters = 98 distance = 3.233388064377964
value : iters = 99 distance = 1.4626442633887822
value : iters = 100 distance = 0.0
```

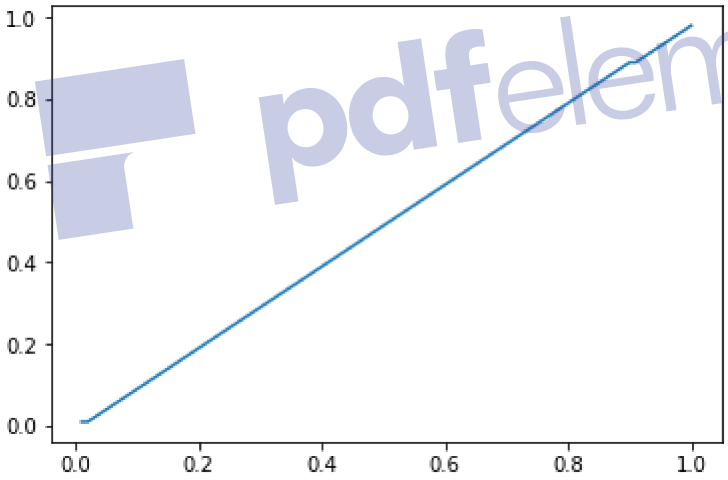
Out[13]: `[<matplotlib.lines.Line2D at 0x1b9b1019940>]`



Thus we see that it takes 100 iterations for convergence with t= T-1 as the starting point.

In [14]:
```
#Problem 5.15
w_dash_vec = w_vec[max_ind]
plt.plot(w_vec, w_dash_vec)
```

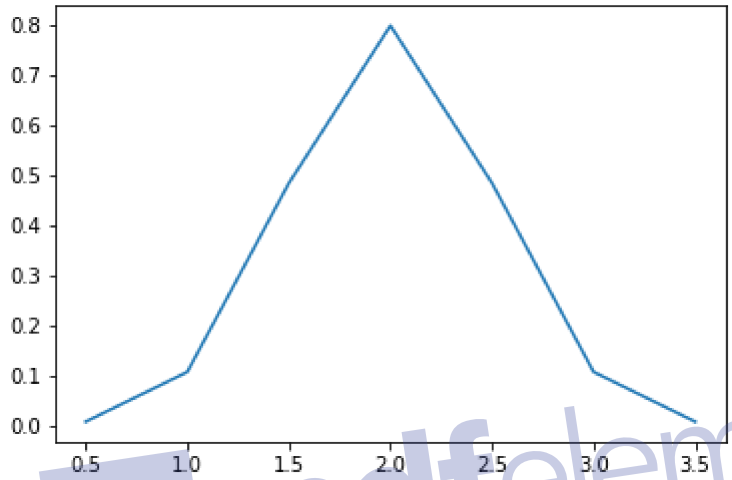
Out[14]:
```
[<matplotlib.lines.Line2D at 0x1b9b1072470>]
```



In [15]:
```
#Problem 5.16
sigma = np.sqrt(0.25)
mu = 4 * sigma
M = 7
eps_vec = np.linspace(mu - 3 * sigma, mu + 3 * sigma, M)
pdf = norm.pdf(eps_vec, loc=mu, scale=sigma)
print(pdf)
plt.plot(eps_vec, pdf)
```

```
[0.0088637  0.10798193 0.48394145 0.79788456 0.48394145 0.10798193
 0.0088637 ]
```

Out[15]: [<matplotlib.lines.Line2D at 0x1b9b209ab70>]

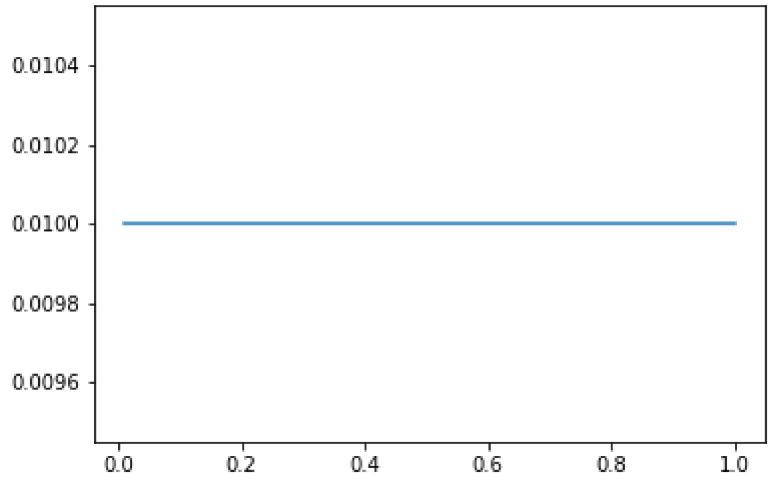


In [16]:
```
#Problem 5.17
c_mat = np.tile(w_vec.reshape(N,1), (1,N)) - np.tile(w_vec.reshape(1,N), (N,1
))
c_pos = c_mat <= 0
c_mat[c_pos] = 1e-10
u_mat = log_utility(c_mat)
ue_vec = np.array([u_mat * i for i in eps_vec])

v_init = np.zeros((N,M))
v_expt = v_init @ pdf.reshape((M,1))
v_expt_mat = np.tile(v_expt.reshape((1,N)), (N,1))
v_expt_mat[c_pos] = -9e+4
v_expt_mat_3D = np.array([v_expt_mat for i in range(M)])
v_t = ue_vec + beta * v_expt_mat_3D
v_new = np.zeros((N, M))
w_dash = np.zeros((N, M))
```

In [17]:
```
for i in range(N):
    v_all = v_t[:, i, :]
    v_new[i] = v_all.max(axis=1)
    max_ind = np.argmax(v_all, axis=1)
    w_dash[i] = w_vec[max_ind]
```

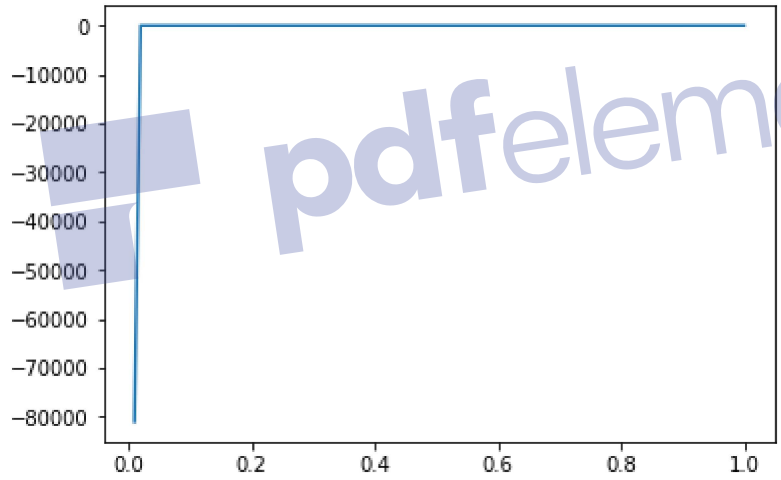
In [18]:
```
plt.plot(w_vec, np.average(w_dash, axis=1))
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x1b9b21046a0>]
```



In [19]:
```
plt.plot(w_vec, np.average(v_new, axis=1))
```

Out[19]:
```
[<matplotlib.lines.Line2D at 0x1b9b2158f28>]
```



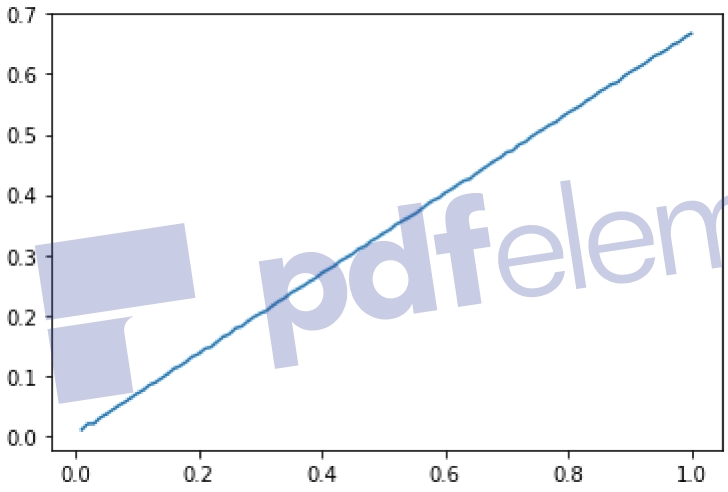
In [20]:
```
#Problem 5.18
dist_1 = sum(sum((v_new - v_init) ** 2))
print(dist_1)
```

```
45979247448.96636
```

In [21]:
```
#Problem 5.19
v_init = v_new.copy()
v_expt = v_init @ pdf.reshape((M,1))
v_expt_mat = np.tile(v_expt.reshape((1,N)), (N,1))
v_expt_mat[c_pos] = -9e+4
v_expt_mat_3D = np.array([v_expt_mat for i in range(M)])
v_t = ue_vec + beta * v_expt_mat_3D
v_new = np.zeros((N, M))
w_dash = np.zeros((N, M))

for i in range(N):
    v_all = v_t[:, i, :]
    v_new[i] = v_all.max(axis=1)
    max_ind = np.argmax(v_all, axis=1)
    w_dash[i] = w_vec[max_ind]
```

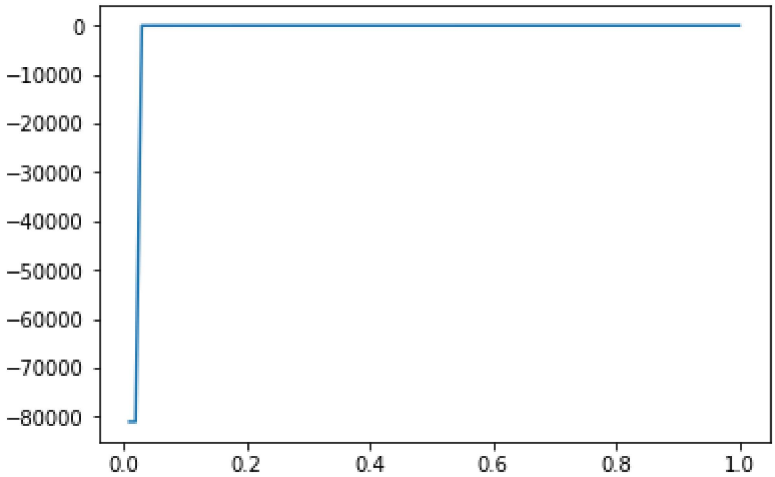
In [22]:
```
plt.plot(w_vec, np.average(w_dash, axis=1))
```

Out[22]:
```
[<matplotlib.lines.Line2D at 0x1b9b21bca58>]
```



In [23]:
```
plt.plot(w_vec, np.average(v_new, axis=1))
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x1b9b2218e10>]
```



In [24]:
```
dist_2 = sum(sum((v_new - v_init) ** 2))
print(dist_2, dist_1)
```

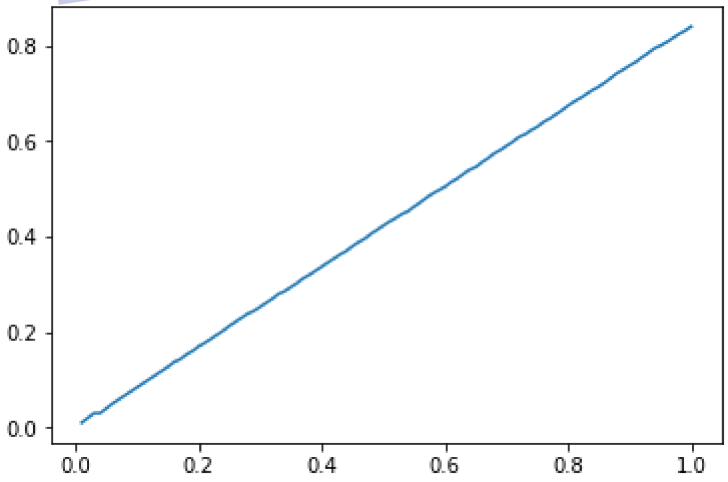
```
45968829677.92325 45979247448.96636
```

We can see that the distance has decreased!

In [25]:
```
#Problem 5.20
v_init = v_new.copy()
v_expt = v_init @ pdf.reshape((M,1))
v_expt_mat = np.tile(v_expt.reshape((1,N)), (N,1))
v_expt_mat[c_pos] = -9e+4
v_expt_mat_3D = np.array([v_expt_mat for i in range(M)])
v_t = ue_vec + beta * v_expt_mat_3D
v_new = np.zeros((N, M))
w_dash = np.zeros((N, M))

for i in range(N):
    v_all = v_t[:, i, :]
    v_new[i] = v_all.max(axis=1)
    max_ind = np.argmax(v_all, axis=1)
    w_dash[i] = w_vec[max_ind]
```

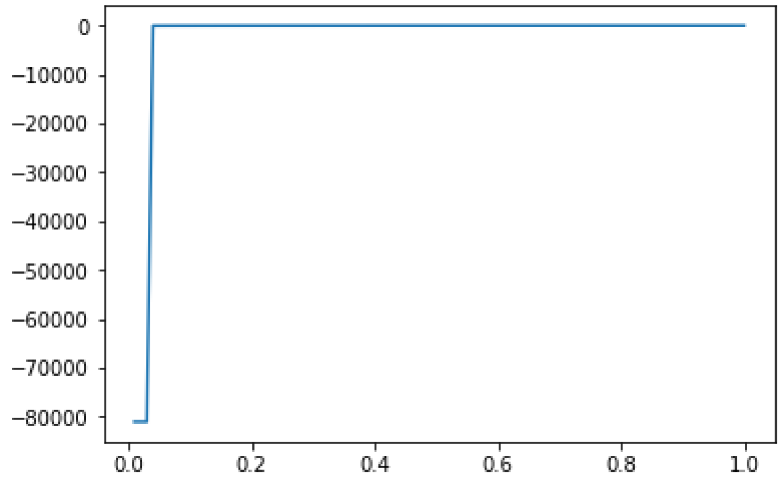
In [26]:
```
plt.plot(w_vec, np.average(w_dash, axis=1))
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x1b9b227ca20>]
```



In [27]:
```
plt.plot(w_vec, np.average(v_new, axis=1))
```

Out[27]:
```
[<matplotlib.lines.Line2D at 0x1b9b22d8278>]
```



In [28]:
```
dist_3 = sum(sum((v_new - v_init) ** 2))
print(dist_3, dist_2, dist_1)
```

```
45950143416.509766 45968829677.92325 45979247448.96636
```

The distance has further reduced!

In [29]:
```
#Problem 5.21
dist = 10
maxiters = 500
toler = 1e-9
vf_iter = 0
v_init = np.zeros((N,M)) #starting from time t=T

while dist > toler and vf_iter < maxiters:
    v_expt = v_init @ pdf.reshape((M,1))
    v_expt_mat = np.tile(v_expt.reshape((1,N)), (N,1))
    v_expt_mat[c_pos] = -9e+4
    v_expt_mat_3D = np.array([v_expt_mat for i in range(M)])
    v_t = ue_vec + beta * v_expt_mat_3D
    v_new = np.zeros((N, M))
    w_dash = np.zeros((N, M))

    for i in range(N):
        v_all = v_t[:, i, :]
        v_new[i] = v_all.max(axis=1)
        max_ind = np.argmax(v_all, axis=1)
        w_dash[i] = w_vec[max_ind]
    dist= sum(sum((v_new - v_init) ** 2))
    v_init = v_new.copy()
    vf_iter += 1
    print("value : iters = {} distance = {}".format(vf_iter, dist))
```

```
value : iters = 1 distance = 45979247448.96636
value : iters = 2 distance = 45968829677.92325
value : iters = 3 distance = 45950143416.509766
value : iters = 4 distance = 45916698826.31781
value : iters = 5 distance = 45857095250.14685
value : iters = 6 distance = 45751696674.42444
value : iters = 7 distance = 45568004744.36679
value : iters = 8 distance = 45256707713.77928
value : iters = 9 distance = 44758728727.70194
value : iters = 10 distance = 44063254831.88394
value : iters = 11 distance = 43458738928.76991
value : iters = 12 distance = 44457224479.50154
value : iters = 13 distance = 52981076462.99518
value : iters = 14 distance = 90008703040.51065
value : iters = 15 distance = 226562184008.26746
value : iters = 16 distance = 457499087150.17194
value : iters = 17 distance = 192392164686.71262
value : iters = 18 distance = 0.0
```

Thus it takes 18 iterations to converge!

In [30]:
```
#Problem 5.22
x,y = np.meshgrid(w_vec, eps_vec)
fig = plt.figure(figsize = (15, 15))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x.T, y.T, w_dash)
ax.set_xlabel('cake today')
ax.set_ylabel('taste shock today')
ax.set_title('cake tomorrow')
ax.view_init(elev = 60, azim = 70)
```