Yuming Liu 1/25/2020

Exercise 1

The condition that characterizes the optimal amount of cake to eat in period 1 is:

$$\max_{W_{T+1}\in[0,W_T]} u(W_T - W_{T+1})$$

Exercise 2

The condition that characterizes the optimal amount of cake to leave for the next period W_3 in period 2 is:

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

The condition that characterizes the optimal amount of cake leave for the next period W_2 in period 1 is:

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

Exercise 3

The condition that characterize the optimal amount of cake to leave for the next period in each period $\{W_2(period1), W_3(period2), W_4(period3)\}$ are:

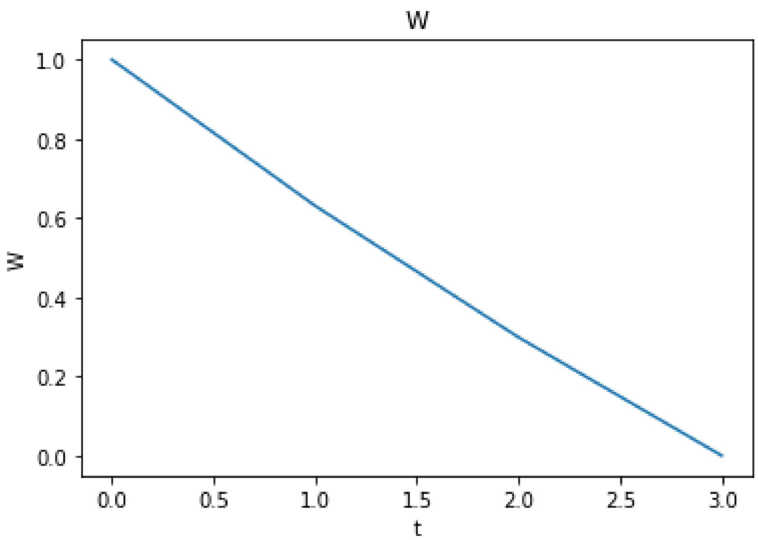
$$\max_{W_2\in[0,W_1]} (u(W_1 - W_2) + \max_{W_3\in[0,W_2]} \beta(u(W_2 - W_3) + \max_{W_4\in[0,W_3]} \beta u(W_3 - W_4)), (1)$$
$$\max_{W_3\in[0,W_2]} (u(W_2 - W_3) + \max_{W_4\in[0,W_3]} \beta u(W_3 - W_4)), (2)$$
$$and \max_{W_4\in[0,W_3]} u(W_3 - W_4)(3).$$

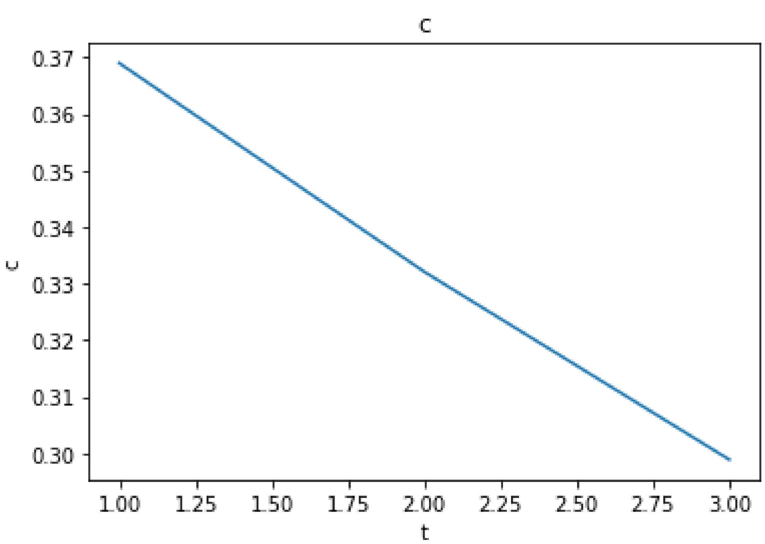
Then we have $W_4 = 0$. To calculate W_2 and W_3, we take the derivatives of (1) and (2) and get

$$0 = u^{'}(1 - W_2) + \beta(u^{'}(W_2 - W_3) + \beta u^{'} W_3)$$
$$0 = u^{'}(W_2 - W_3) + \beta u^{'} W_3$$

Therefore, since we know $\beta = 0.9$ and the period utility function is $\ln(c_t)$, we have $W_2 = 0.631$ and $W_3 = 0.299$. Then we get $c_1 = 0.299$, $c_2 = 0.332$, and $c_3 = 0.299$. How $\{c_t\}_{t=1}^{3}$ and ${Wt}^4{t=1} evolve over the three periods could be shown as:

In [1]:
```
import matplotlib.pyplot as plt

beta = 0.9
W = [1, 1-1/(1+beta+beta**2),1-(1+beta)/(1+beta+beta**2),0]
c = [(W[i]-W[i+1]) for i in range(len(W)-1)]
t = [0,1,2,3]

plt.plot(t,W)
plt.title("W")
plt.xlabel("t")
plt.ylabel("W")
plt.show()

plt.plot(t[1:],c)
plt.title("c")
plt.xlabel("t")
plt.ylabel("c")
plt.show()
```





Exercise 4

Taking the derivative of the function, we get that

$$-u^{'}(W_{T-1} - \psi_{T-1}(W_{T-1})) + \beta u^{'}(\psi_{T-1}(W_{T-1})) = 0.$$

Therefore, the funtion V_{T-1} could be written as

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

Exercise 5

We have $V_{T-1}(\bar{W}) = u(\bar{W} - \psi_{T-1}\bar{W})) + \beta u(\psi_{T-1}(\bar{W}))$. Then taking the derivative, we have $-u^{'}(\bar{W} - \psi_{T-1}(\bar{W})) + \beta u^{'}(\psi_{T-1}(\bar{W})) = 0$. Then we get $\psi_{T-1}(\bar{W}) = \frac{\beta}{1+\beta}\bar{W}$.

We get $V_{T-1}(\bar{W}) = \ln((1 - \frac{\beta}{1+\beta})\bar{W}) + \beta\ln(\frac{\beta}{1+\beta}\bar{W}) = \ln(\frac{1}{1+\beta}\bar{W}) + \beta\ln(\frac{\beta}{1+\beta}\bar{W})$. Since $V_T(\bar{W}) = u(\bar{W}) = \ln(\bar{W})$ and $\psi_T(\bar{W}) = 0$, we have $V_T(\bar{W}) \neq V_{T-1}(\bar{W})$ and $\psi_T(\bar{W}) \neq \psi_{T-1}(\bar{W})$.

Exercise 6

We have the finite horizon Bellman equation for the value function at time $T-2$ is

$$\begin{aligned}
V_{T-2}(W_{T-2}) &\equiv \max_{W_{T-1}} \ln(W_{T-2} - W_{T-1}) + \beta V_{T-1}(W_{T-1}) \\
&\equiv \max_{W_{T-1}} \ln(W_{T-2} - W_{T-1}) + \beta(\ln(\frac{1}{1+\beta}W_{T-1}) + \beta\ln(\frac{\beta}{1+\beta}W_{T-1})) \\
&\equiv \max_{W_{T-1}} \ln(W_{T-2} - W_{T-1}) + \beta\ln(\frac{1}{1+\beta}W_{T-1}) + \beta^2\ln(\frac{\beta}{1+\beta}W_{T-1}) \\
&\equiv \max_{W_{T-1}} \ln(W_{T-2} - W_{T-1}) + (\beta+\beta^2)\ln(W_{T-1}) + \beta\ln(\frac{1}{1+\beta}) + \beta^2\ln(\frac{\beta}{1+\beta})
\end{aligned}$$

Since we have $W_{T-1} = \psi_{T-2}(W_{T-2})$, we take the derivative and get

$$-\frac{1}{W_{T-2} - \psi_{T-2}(W_{T-2})} + \frac{\beta+\beta^2}{\psi_{T-2}(W_{T-2})} = 0.$$

Therefore, the analytical solutions are

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

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

Exercise 7

From exercise 5 and 6, we find that $\psi_{T-s}(W_{T-s}) = (1-\frac{1}{\sum_{i=0}^{s}\beta^i})W_{T-s}$ and $V_{T-s}(W_{T-s}) = \sum_{i=0}^{s}\beta^i\ln(\frac{\beta^i}{\sum_{i=0}^{s}\beta^i}W_{T-s})$.

Then we have $\lim_{s\to\infty}\psi_{T-s}(W_{T-s}) = \beta W_{T-s} = \psi(W_{T-s})$ and $\lim_{s\to\infty}V_{T-s}(W_{T-s}) = \frac{1}{1-\beta}$

Exercise 8

When the horizon is infinite, we have the Bellman equation for the problem is

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

Exercise 9

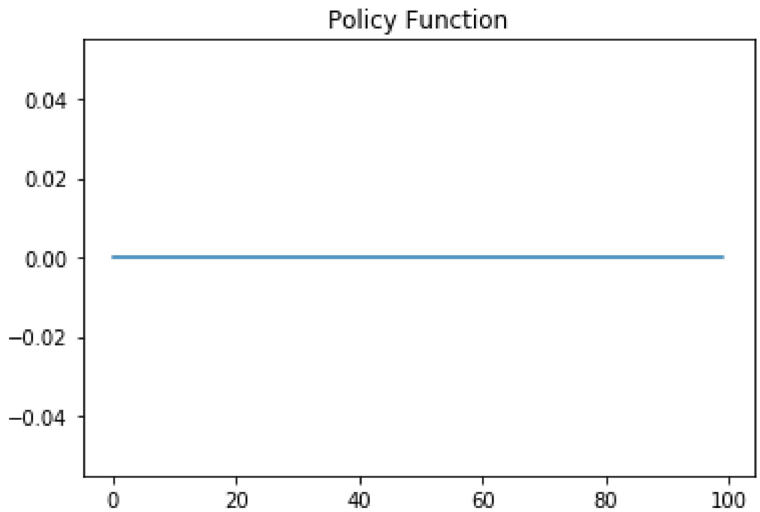
In [2]:
```
import numpy as np

W = np.linspace(0.01, 1, 100)
```

Exercise 10

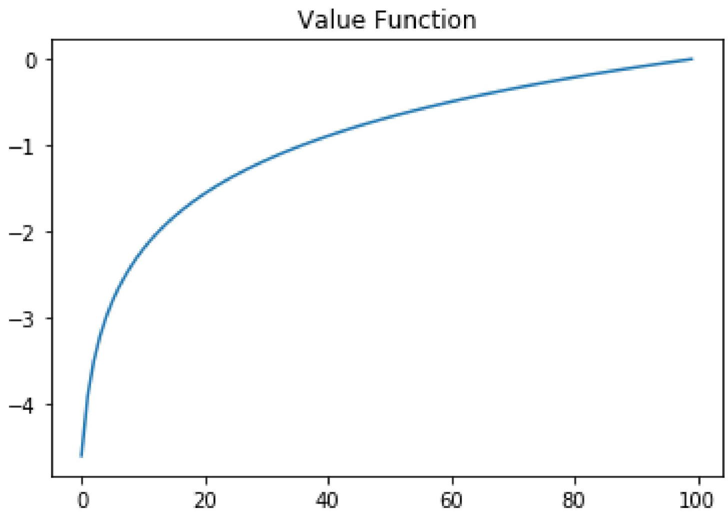
In [3]:
```
def u(c):
    new_c = np.log(c)
    return new_c

beta = 0.9
```

In [4]:
```
W0 = np.zeros(100)
V0 = np.zeros(100)
u_p = u(W-W0)
V_p = u_p+beta*W0
```

In [5]:
```
plt.plot(W0)
plt.title("Policy Function")
plt.show()
```



In [6]:
```
plt.plot(V_p)
plt.title("Value Function")
plt.show()
```



Exercise 11

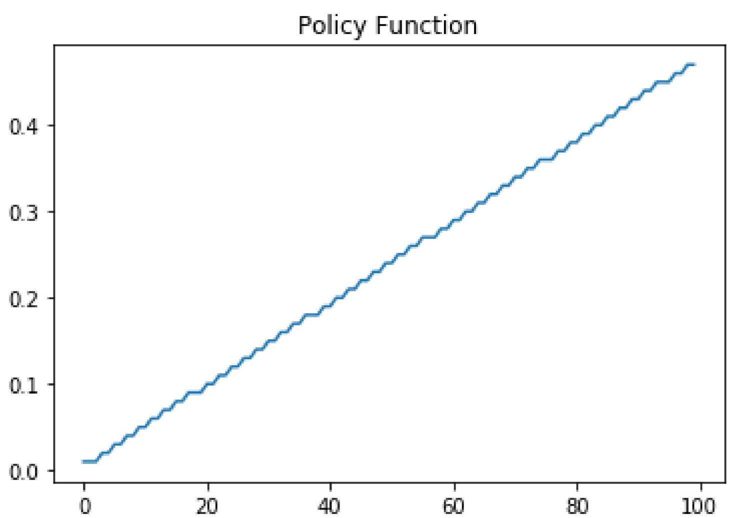
In [7]:
```
V_Tp1 = np.zeros(100)
dist = np.sum((V_p-V_Tp1)**2)
print('The distance metric is ', dist)
```

```
The distance metric is  178.92611065972804
```

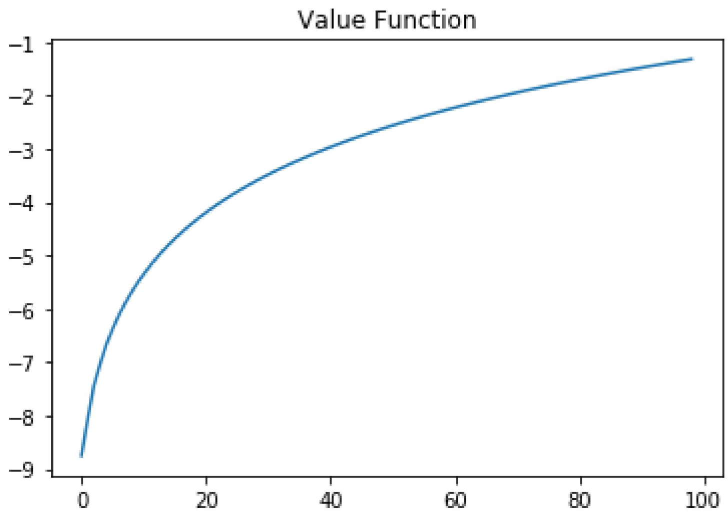
Exercise 12

In [8]:
```
c_mat = W.reshape(-1,1)-W
c_pos = c_mat > 0
c_mat[~c_pos] = 1e-10
u_mat = u(c_mat)

V = np.tile(V_p.reshape((1,100)),(100,1))
V[~c_pos] = -9e+4
V_Tm1 = (u_mat+beta*V).max(axis = 1)
W_index = np.argmax(u_mat+beta*V,axis = 1)
W_T = W[W_index]
```

In [9]:
```
plt.plot(W_T)
plt.title("Policy Function")
plt.show()
```



In [10]:
```
plt.plot(V_Tm1[1:])
plt.title("Value Function")
plt.show()
```



In [11]:
```
dist = np.sum((V_p-V_Tm1)**2)
print('The distance metric is ', dist)
```

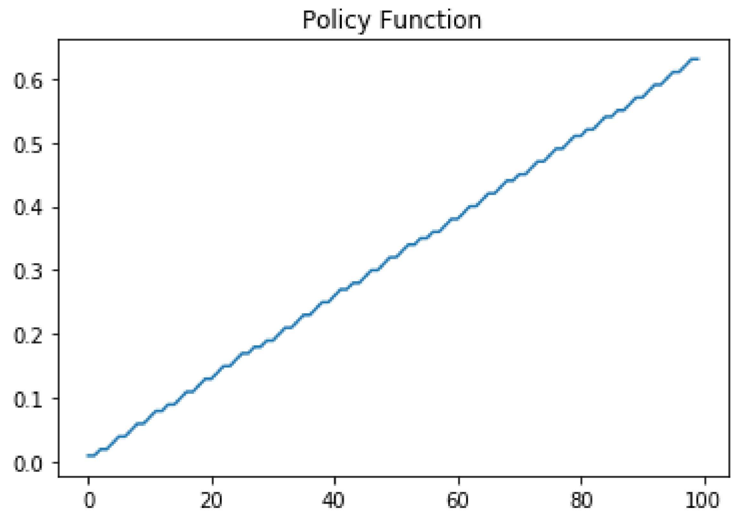
```
The distance metric is  6563985007.657785
```

The distance is larger than the previous one.

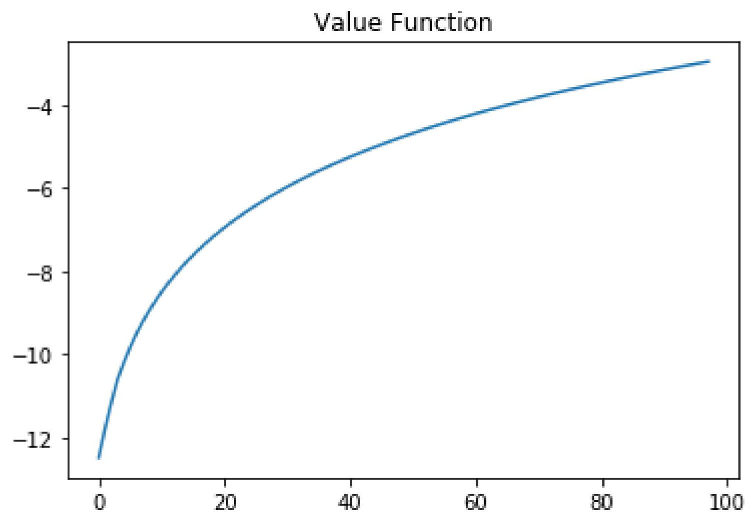
Exercise 13

In [12]:
```
c_mat = W.reshape(-1,1)-W
c_pos = c_mat > 0
c_mat[~c_pos] = 1e-10
u_mat = u(c_mat)

V = np.tile(V_Tm1.reshape((1,100)),(100,1))
V[~c_pos] = -9e+4
V_Tm2 = (u_mat+beta*V).max(axis = 1)
W_index = np.argmax(u_mat+beta*V,axis = 1)
W_T = W[W_index]
```

In [13]:
```
plt.plot(W_T)
plt.title("Policy Function")
plt.show()
```



In [14]:
```
plt.plot(V_Tm2[2:])
plt.title("Value Function")
plt.show()
```



In [15]:
```
dist = np.sum((V_Tm1-V_Tm2)**2)
print('The distance metric is ', dist)
```

```
The distance metric is  5316828035.491551
```

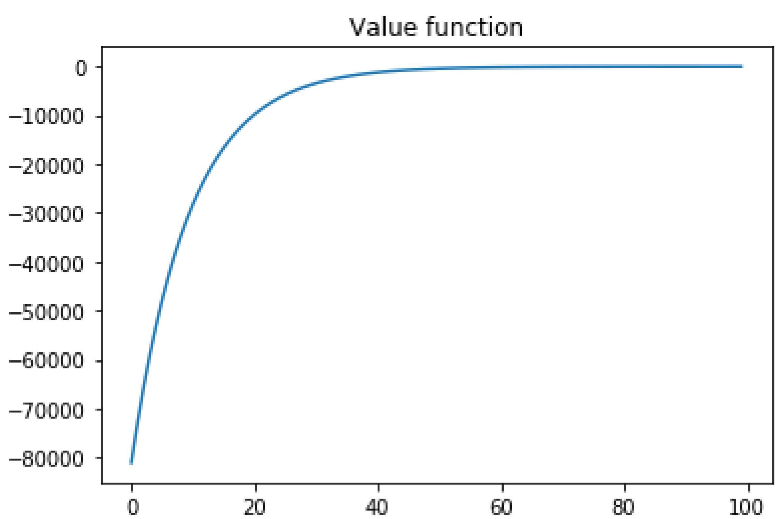
The distance is smaller than the previous one.

Exercise 14

In [16]:
```
c_mat = W.reshape(-1,1)-W
c_pos = c_mat > 0
c_mat[~c_pos] = 1e-10
u_mat = u(c_mat)

min_dist = 1e-10
Iter = 0
stop_iter = 1000
V0 = u_p

while Iter < stop_iter and dist > min_dist:
    V = np.tile(V0.reshape((1,100)),(100,1))
    V[~c_pos] = -9e+4
    new_V = (u_mat+beta*V).max(axis = 1)
    dist = np.sum((new_V-V0)**2)
    V0 = new_V
    W_index = np.argmax(u_mat+beta*V,axis = 1)
    Iter += 1

print(Iter, dist)
print("After "+str(Iter)+" iterations, the function converges to the fixed poi
nt.")
```

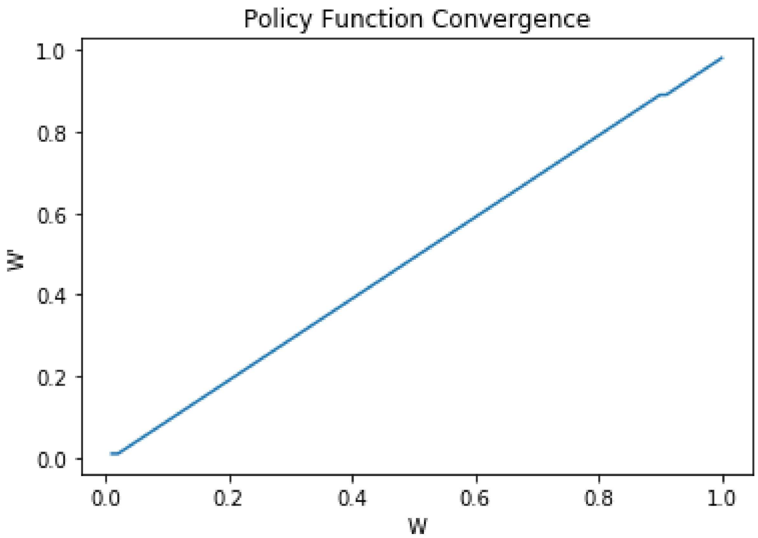
```
101 0.0
After 101 iterations, the function converges to the fixed point.
```

In [17]:
```
plt.plot(V0)
plt.title("Value function")
plt.show()
```



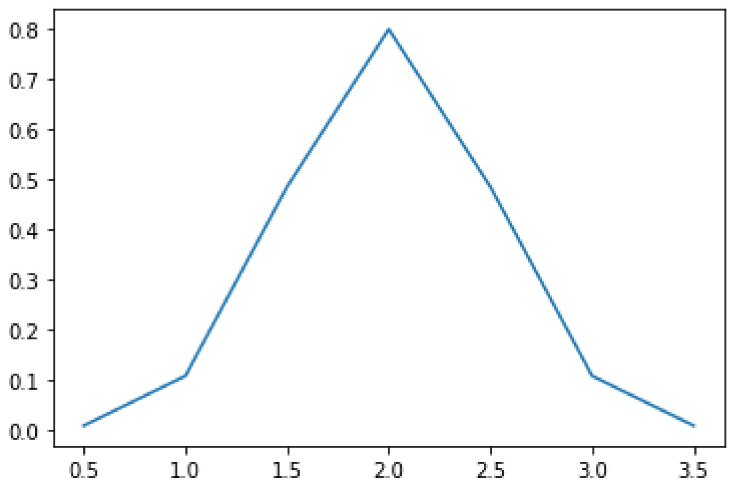
Exercise 15

In [18]:
```
W_T = W[W_index]
plt.plot(W,W_T)
plt.title("Policy Function Convergence")
plt.xlabel("W")
plt.ylabel("W'")
plt.show()
```



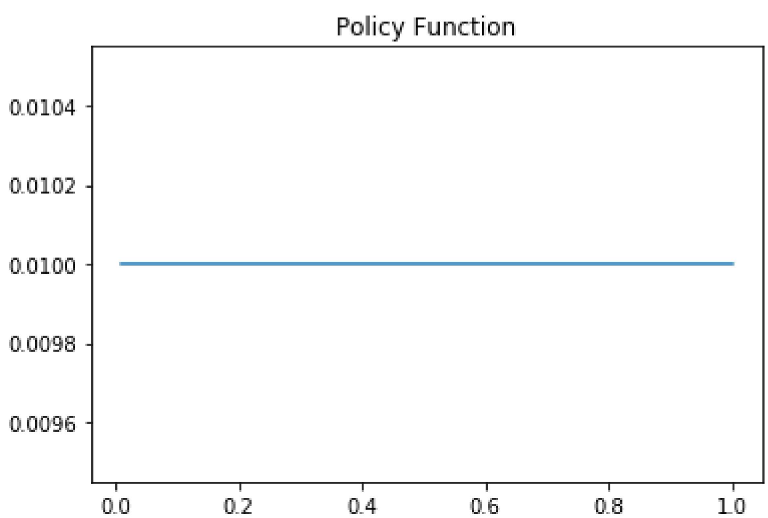
Exercise 16

In [19]:
```
from scipy.stats import norm

sigma = 0.5
mu = 4*sigma
M = 7
epsilon = np.linspace(mu-3*sigma,mu+3*sigma,M)
PDF = lambda x: norm(loc = mu, scale = sigma).pdf(x)
pdf = PDF(epsilon)
plt.plot(epsilon, pdf)
plt.show()
```

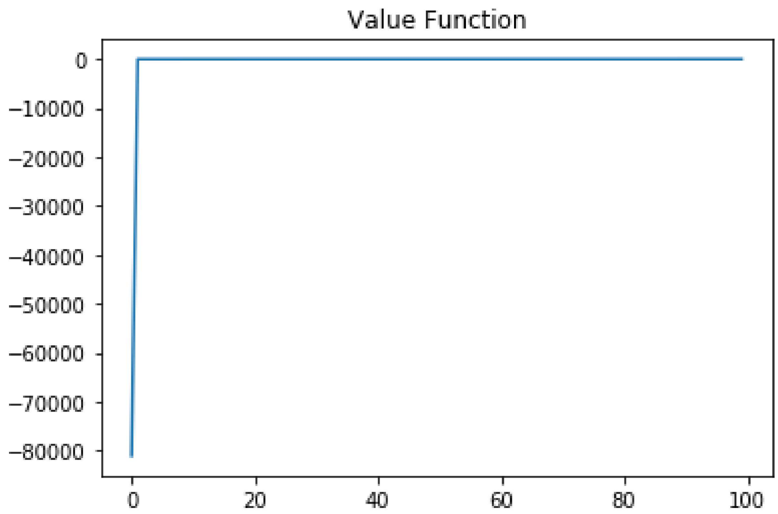


Exercise 17

In [20]:
```
c_mat = W.reshape(-1,1)-W
c_pos = c_mat > 0
c_mat[~c_pos] = 1e-10
u_mat = u(c_mat)

d3dim = np.array([u_mat*e for e in epsilon])

V0 = np.zeros((100, M))
EV = V0 @ pdf.reshape((M,1))
EV_mat = np.tile(EV.reshape((1,100)),(100,1))
EV_mat[~c_pos] = -9e+4
EV_3d = np.array([EV_mat for i in range(M)])

V_3d = d3dim + beta*EV_3d
V_new = np.zeros((100,M))
W_prime = np.zeros((100,M))

for i in range(100):
    array = V_3d[:, i, :]
    V_new[i] = array.max(axis=1)
    W_index = np.argmax(array, axis=1)
    W_prime[i] = W[W_index]

plt.plot(W,np.average(W_prime ,axis=1))
plt.title("Policy Function")
plt.show()
```



In [21]:
```
plt.plot(np.average(V_new, axis=1))
plt.title("Value Function")
plt.show()
```



Exercise 18

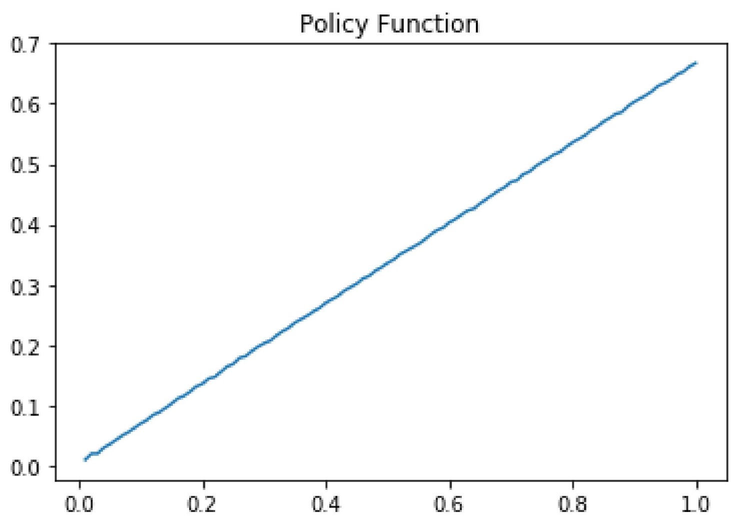
In [22]:
```
V_Tm1 = np.zeros((100,M))
dist = np.sum((V_new-V_Tm1)**2)
print('The distance metric is ', dist)
```

```
The distance metric is  45979247448.96637
```

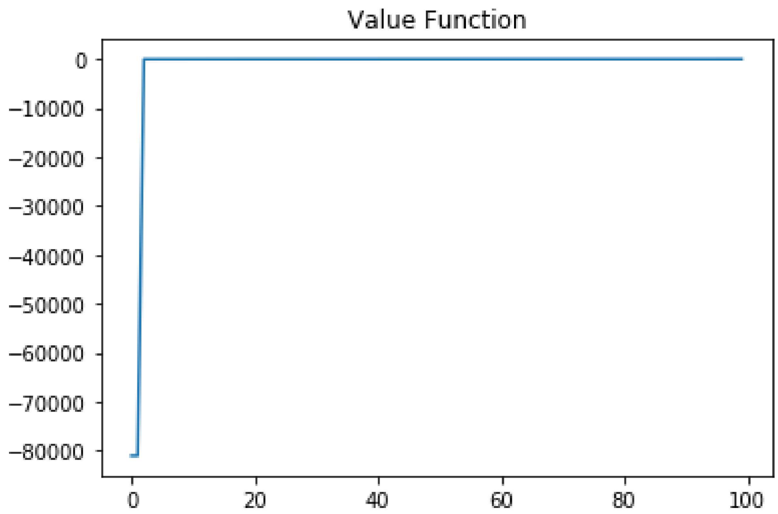
Exercise 19

In [23]:

```
c_mat = W.reshape(-1,1)-W
c_pos = c_mat > 0
c_mat[~c_pos] = 1e-10
u_mat = u(c_mat)

d3dim = np.array([u_mat*e for e in epsilon])

V0 = V_new
EV = V0 @ pdf.reshape((M,1))
EV_mat = np.tile(EV.reshape((1,100)),(100,1))
EV_mat[~c_pos] = -9e+4
EV_3d = np.array([EV_mat for i in range(M)])

V_3d = d3dim + beta*EV_3d
V_new = np.zeros((100,M))
W_prime = np.zeros((100,M))

for i in range(100):
    array = V_3d[:, i, :]
    V_new[i] = array.max(axis=1)
    W_index = np.argmax(array, axis=1)
    W_prime[i] = W[W_index]

plt.plot(W,np.average(W_prime ,axis=1))
plt.title("Policy Function")
plt.show()
```



In [24]:
```
plt.plot(np.average(V_new, axis=1))
plt.title("Value Function")
plt.show()
```



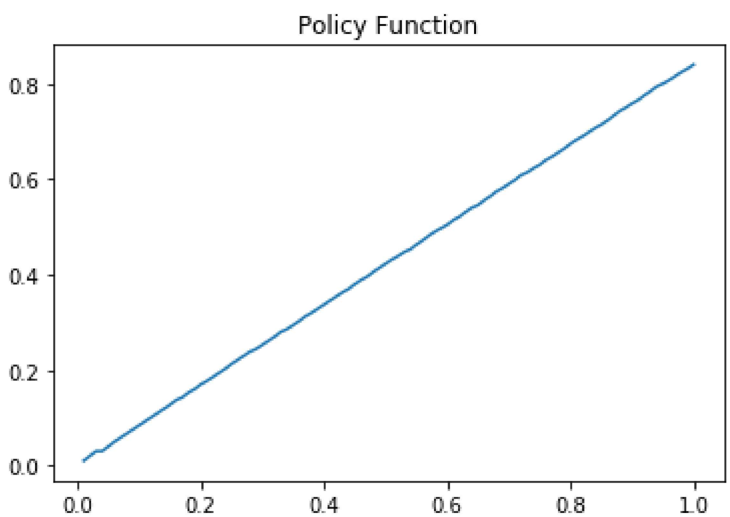
In [25]:
```
dist = np.sum((V_new-V0)**2)
print('The distance metric is ', dist)
```

```
The distance metric is  45968829677.923256
```

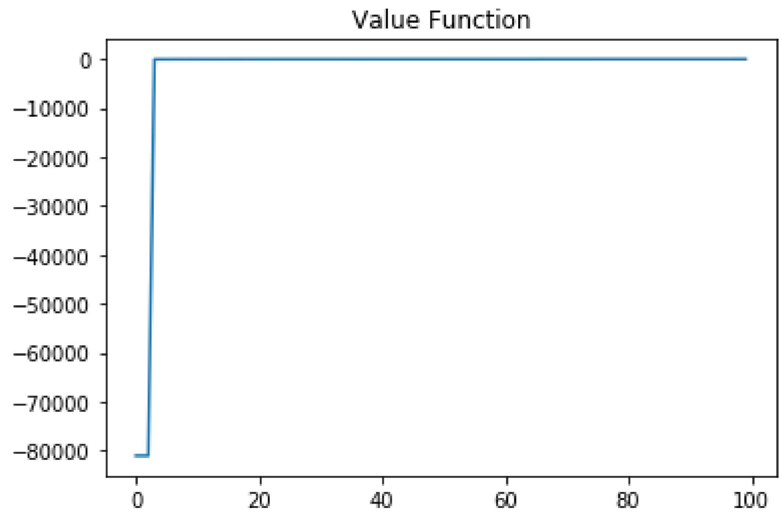
The distance is smaller than the previous one.

Exercise 20

In [26]:
```
c_mat = W.reshape(-1,1)-W
c_pos = c_mat > 0
c_mat[~c_pos] = 1e-10
u_mat = u(c_mat)

d3dim = np.array([u_mat*e for e in epsilon])

V0 = V_new
EV = V0 @ pdf.reshape((M,1))
EV_mat = np.tile(EV.reshape((1,100)),(100,1))
EV_mat[~c_pos] = -9e+4
EV_3d = np.array([EV_mat for i in range(M)])

V_3d = d3dim + beta*EV_3d
V_new = np.zeros((100,M))
W_prime = np.zeros((100,M))

for i in range(100):
    array = V_3d[:, i, :]
    V_new[i] = array.max(axis=1)
    W_index = np.argmax(array, axis=1)
    W_prime[i] = W[W_index]

plt.plot(W,np.average(W_prime ,axis=1))
plt.title("Policy Function")
plt.show()
```



In [27]:
```
plt.plot(np.average(V_new, axis=1))
plt.title("Value Function")
plt.show()
```



In [28]:
```
dist = np.sum((V_new-V0)**2)
print('The distance metric is ', dist)
```

```
The distance metric is  45950143416.50976
```

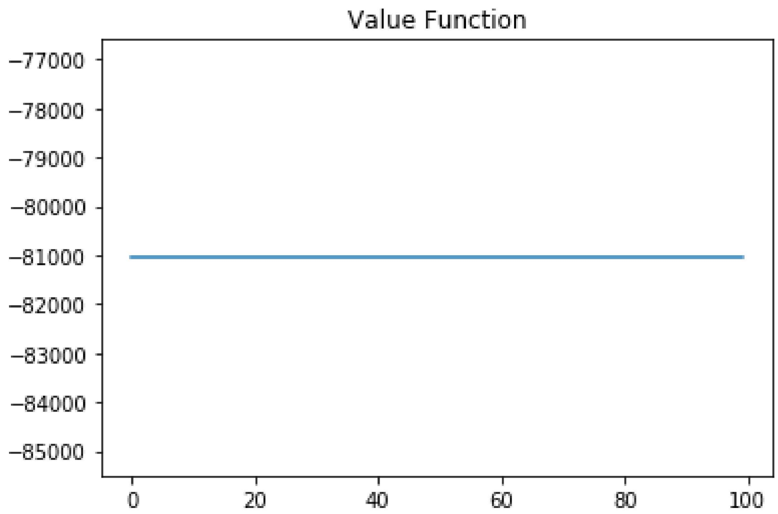
The distances keep decreasing through δ_T to δ_{T-2}.

Exercise 21

In [29]:
```
c_mat = W.reshape(-1,1)-W
c_pos = c_mat > 0
c_mat[~c_pos] = 1e-10
u_mat = u(c_mat)

min_dist = 1e-9
Iter = 0
stop_iter = 1000
V0 = np.zeros((100,M))

while dist > min_dist and Iter < stop_iter:
    EV = V0 @ pdf.reshape((M,1))
    EV_mat = np.tile(EV.reshape((1,100)),(100,1))
    EV_mat[~c_pos] = -9e+4
    EV_3d = np.array([EV_mat for i in range(M)])

    V_3d = d3dim + beta*EV_3d
    V_new = np.zeros((100,M))
    W_prime = np.zeros((100,M))

    for i in range(100):
        array = V_3d[:, i, :]
        V_new[i] = array.max(axis=1)
        W_index = np.argmax(array, axis=1)
        W_prime[i] = W[W_index]

    dist = np.sum((V_new-V0)**2)

    V0 = V_new
    Iter += 1

print(Iter, dist)
print("After "+str(Iter)+" iterations, the function converges to the fixed poi
nt.")
```

```
18 0.0
After 18 iterations, the function converges to the fixed point.
```

In [30]:
```
plt.plot(np.average(V0, axis=1))
plt.title("Value Function")
plt.show()
```



Exercise 22

In [31]:
```
from mpl_toolkits.mplot3d import Axes3D

X, Y = np.meshgrid(W, epsilon)
new_fig = plt.figure(figsize=(10,10))
new_plot = new_fig.add_subplot(111, projection='3d')
new_plot.plot_surface(X.T, Y.T, W_prime)
new_plot.set_xlabel('cake today')
new_plot.set_ylabel('taste shock today')
new_plot.set_zlabel('cake tomorrow')
new_plot.set_title('Policy Function Convergence')
new_plot.view_init(elev=50,azim=50)
plt.show()
```