In [4]:
```python
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from random import seed
from random import random
seed(57)

#set seed for repeatability
np.random.seed(314)
```

In [5]:
```python
#acquire data from CSV
usm_raw_data = pd.read_csv("C:/Users/Taylor_Vitunic/Desktop/Machine Learning/
usm_raw_data.head(10)
```

Out[5]:

	flatness_ratio	symmetry	crossflow	flow_velocity_path_1	flow_velocity_path_2	flow_velocity
0	0.841499	1.009367	0.993816	8.469805	10.178717	1
1	0.841150	1.006584	0.996605	7.531891	9.139914	
2	0.840713	1.011647	0.998152	6.641699	7.975464	
3	0.841119	1.017807	0.996812	5.687514	6.814334	
4	0.840358	1.016534	0.996111	5.660385	6.819560	
5	0.838901	1.014557	0.995404	5.646000	6.830114	
6	0.841544	1.010160	0.995604	5.618586	6.811160	
7	0.840916	1.015113	0.995890	5.647400	6.811408	
8	0.841671	1.008904	0.994111	5.613118	6.815111	
9	0.835900	1.014731	0.997580	4.744331	5.749815	

10 rows × 37 columns

In [6]:
```python
#convert data
usm_data = usm_raw_data.to_numpy()
```

In [7]:
```python
#check shape
usm_data.shape
```

Out[7]: (87, 37)

In [8]:
```python
#get USM data into workable numpy

A=[]

for i in range(0,87):
    C=[]
    for j in range(0,37):
        C.append(usm_data[i,j])
    A.append(C)

A = np.array(A)

#shuffle array since it is ordered by failure
np.random.shuffle(A)
```

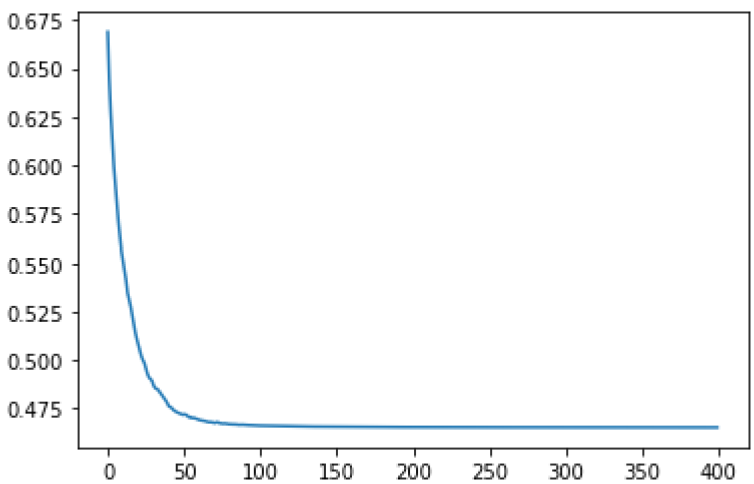
In [9]:
```python
#normalize data to make learning easier

#def normalize(num, avg, stddev):
#    return (num-avg)/stddev

description = usm_raw_data.describe()
avg = np.array(description.T['mean'])
std = np.array(description.T['std'])

for i in range(0,87):
    for j in range(0,36):
        A[i,j] = (A[i,j] - avg[j])/std[j]
```

In [10]:
```python
#build neural network, relu for speed+accuracy, sigmoid at end to ensure answ

model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation = "relu"),
    tf.keras.layers.Dense(128, activation = "relu"),
    tf.keras.layers.Dense(1, activation = "sigmoid"),
])

model.compile(
    loss = tf.keras.losses.BinaryCrossentropy(from_logits=True),
    optimizer = "adam",
    metrics = ["accuracy"]
)
```

In [11]:

```
#run neural network

history = model.fit(A[:70,:36], A[:70,36], epochs=400, verbose=0, shuffle=Tru
plt.plot(history.history["loss"])
```

Out[11]: [<matplotlib.lines.Line2D at 0x25b632315c8>]



In [12]:

```
#predict working/faulty from test set
predictions = model.predict(A[70:,:36])
predictions = [1 if p >=.5 else 0 for p in predictions]
print(predictions)
```

```
[0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0]
```

In [13]:

```
#actuals working/faulty in the test set
actuals = A[70:,36]
print(actuals)
```

```
[0. 0. 1. 1. 0. 0. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 0.]
```

In [14]:

```
#take stock of accuracy of predictions
correct = 0
t1 = 0
t2 = 0
total = len(predictions)

#correct is correctly predicted if meter is faulty or not
#type 1 error = predicting meter is faulty when it is not
#type 2 error = predicting meter is not faulty when it is

for i in range(0, total):

    if (predictions[i]==actuals[i]):
        correct += 1
    elif (predictions[i]>=actuals[i]):
        t1 += 1
    else:
        t2 += 1
print("We got {}/{} correct with {}/{} type 2 errors and {}/{} type 1 errors"
```

```
We got 13/17 correct with 0/17 type 2 errors and 4/17 type 1 errors
```