7.1

Introduction

With N = 1000, do the Fourier transform of these functions:

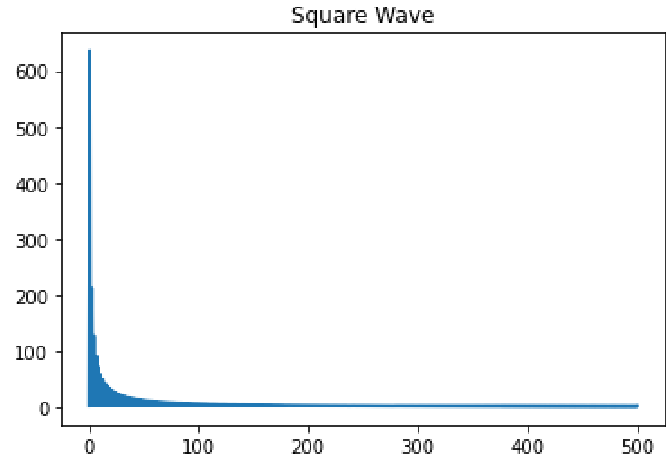
A) Square wave function, single cycle

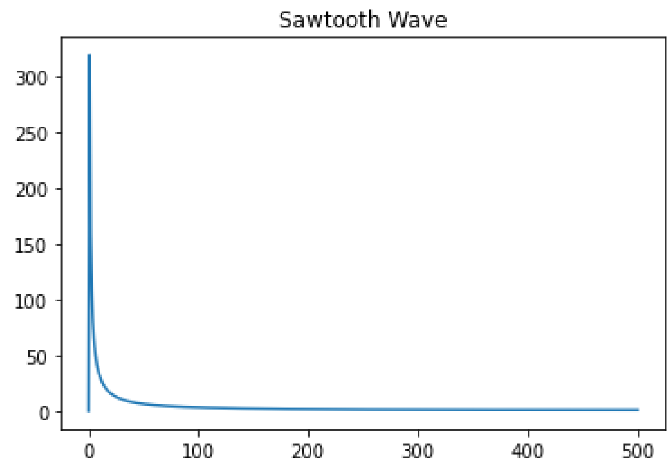
B) Sawtooth wave $y_n = n$

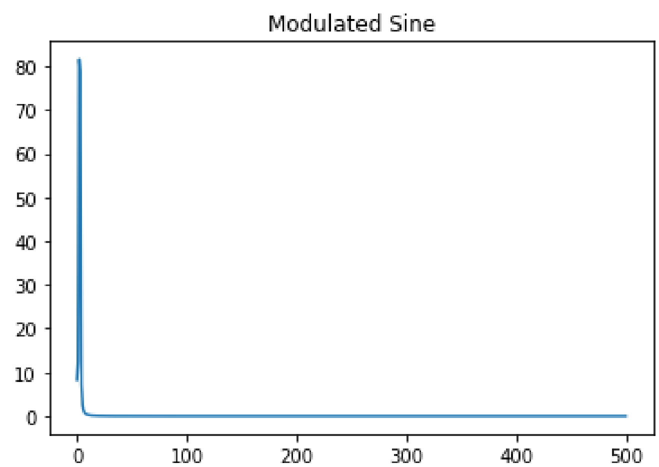
C) Modulated sine wave $y_n = sin(\pi n/N)sin(20\pi n/N)$

In [49]:

```python
import numpy as np
import matplotlib.pyplot as plt
import cmath as cm


def dft(y):
    N = len(y)
    c = np.zeros(N//2 + 1,complex)
    for k in range(N//2+1):
        for n in range(N):
            c[k] += y[n]*cm.exp(-2j*cm.pi*k*n/N)
    return c

#Part A
def square_wave(x):
    if np.sign(np.sin(2*np.pi*x)) > 0:
        return 1
    else:
        return -1
N = 1000
x = np.linspace(0,1,N)
Square_Wave = np.vectorize(square_wave)
y = Square_Wave(x)
c = dft(y)
plt.plot(abs(c))
plt.title('Square Wave')
plt.show()

#Part B
def sawtooth(X):
    return x
x = np.linspace(-1,1,N)
y = sawtooth(x)
c = dft(y)
plt.plot(abs(c))
plt.title('Sawtooth Wave')
plt.show()

#Part C
x = np.linspace(-20*2*np.pi,20*2*np.pi,N)
def mod_sine(x):
    return np.sin(np.pi*x/N)*np.sin(20*np.pi*x/N)
y = mod_sine(x)
c = dft(y)
plt.plot(abs(c))
plt.title('Modulated Sine')
plt.show()
```







Conclusion

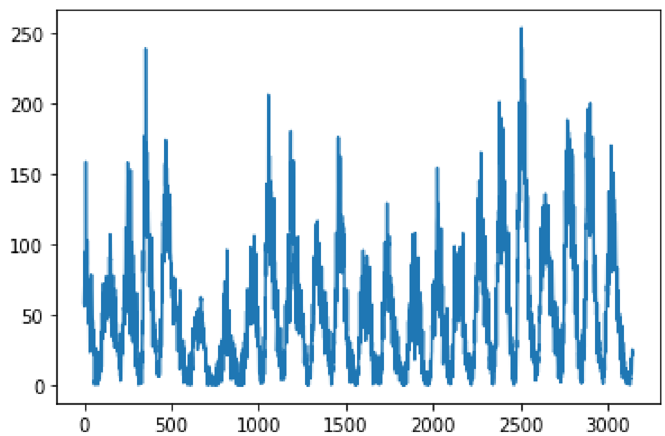
This is hard for me. I cannot get a feel for whether or not it is correct. I was stuck on how it shot up at 0, but now I am seeing that it is just the integral of it. But that doesn't make a ton of sense to me because I thought it would be the average number.

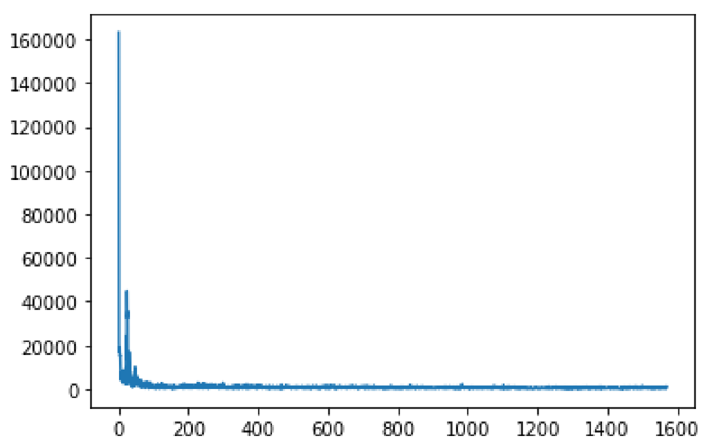
7.2

Introduction

In [36]:

```
import numpy as np
import matplotlib.pyplot as plt
import cmath as cm

def dft(y):
    N = len(y)
    c = np.zeros(N//2 + 1,complex)
    for k in range(N//2+1):
        for n in range(N):
            c[k] += y[n]*cm.exp(-2j*cm.pi*k*n/N)
    return c

data = np.loadtxt("sunspots.txt")[:,1]
plt.plot(data)
plt.show()
c = dft(data)
plt.plot(abs(c))
#plt.xlim(0,500)
plt.show()
print(f'The corresponding peak value is at k ={list(c).index(max(c[1:]))}')
```






```
The corresponding peak value is at k =26
```

Conclusion

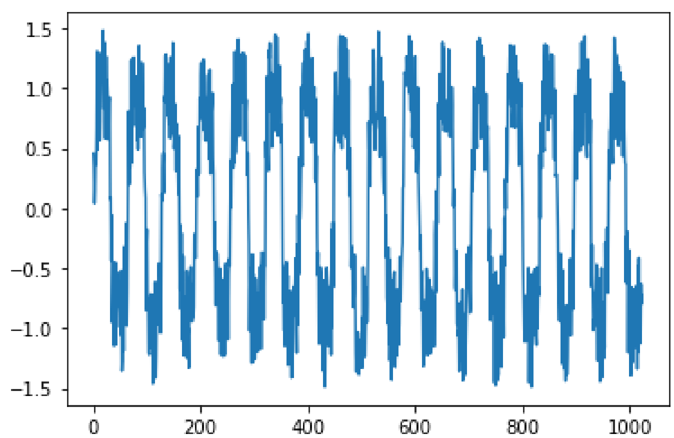
I am not sure if this is correct. But I thought it was fun.

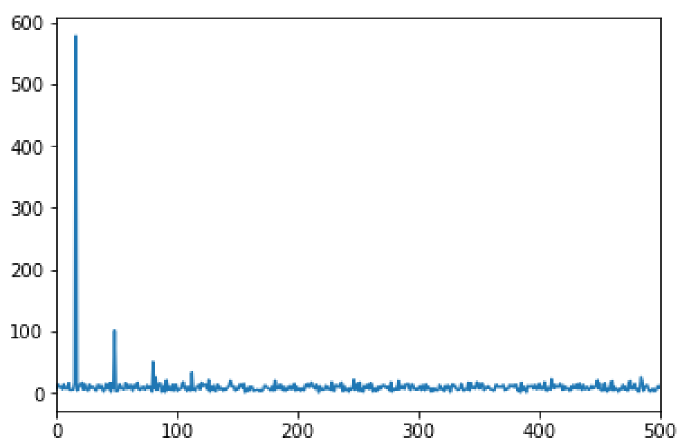
Fourier Transform In Class

Introduction

I am doing it for the pitch.txt file on Newman's website

In [33]:
```
import numpy as np
import matplotlib.pyplot as plt
import cmath as cm

def dft(y):
    N = len(y)
    c = np.zeros(N//2 + 1,complex)
    for k in range(N//2+1):
        for n in range(N):
            c[k] += y[n]*cm.exp(-2j*cm.pi*k*n/N)
    return c

data = np.loadtxt("pitch.txt")
plt.plot(data)
plt.show()
c = dft(data)
plt.plot(abs(c))
plt.xlim(0,500)
plt.show()
```





Conclusion

I thought this was easy. It helped we did it in class and I could see that it was working.