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

Question - 1

The package used in this notebook is numpy.fft and evaluates the DFT as per equation (1) given below:

$$c_k = \sum_{j=0}^{N-1} f_j e^{-i2\pi kj/N} \tag{1}$$

Question - 2

Th inverse DFT computed by the fft package is as follows:

$$f_j = \frac{1}{N}\sum_{k=0}^{N-1} c_k e^{-i2\pi kj/N} \tag{2}$$

Question - 4

Using Euler's Formula:

$$e^{ix} = cos(x) + isin(x) \tag{3}$$

Equation (1) can therefore be written as:

$$c_k = \sum_{k=0}^{N-1} f_j e^{-i2\pi kj/N} = \sum_{k=0}^{N-1} f_j \left[cos\left(\frac{-2\pi kj}{N}\right) + isin\left(\frac{-2\pi kj}{N}\right)\right] \tag{4}$$

Since:

$$x = j\frac{2\pi}{N}$$

Equation (4) can be written as:

$$c_k = \sum_{k=0}^{N-1} f_j (cos(xj) - isin(xj)) \tag{5}$$

Therefore, the coefficients found by the package c_k are related to the ones in $P_N(x)$ as follows:

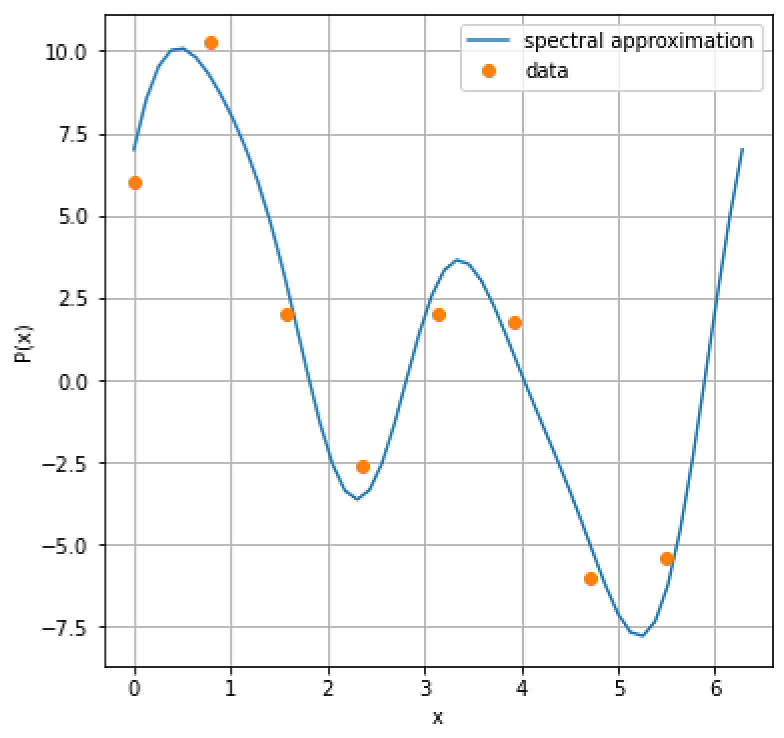
$$c_k = \frac{N}{2}a_k - i\frac{N}{2}b_k$$

So, a_k is the real part of the coefficients computed using np.fft package mutiplied by a factor of $2/N$ and b_k is the imaginary part of the coefficients computed using np.fft package mutiplied by a factor of $2/N$.

Question - 5

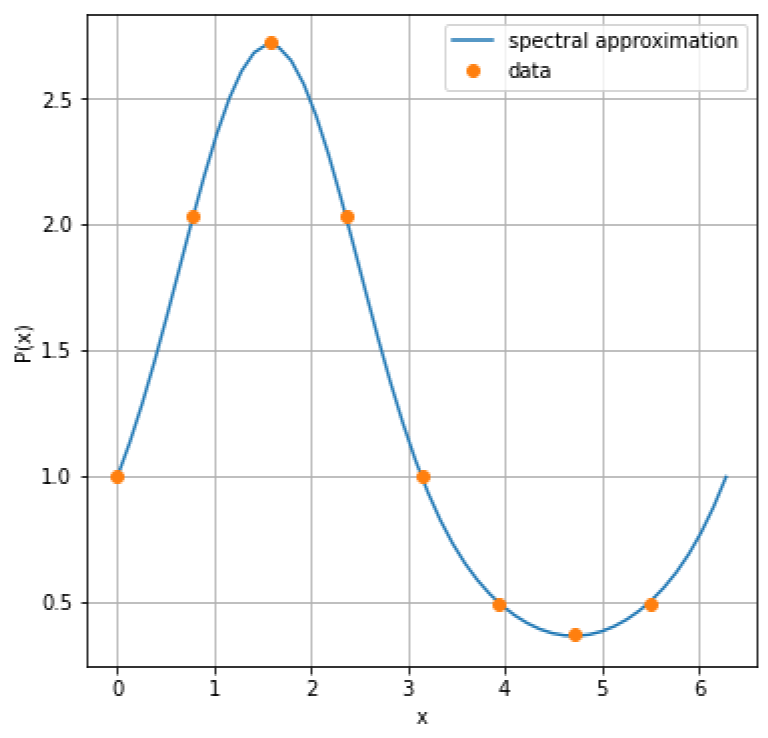
In [2]:
```python
def find_PN(f, xj, x):
    N = len(f)
    ck = np.fft.fft(f)
    a = (2/N)*np.real(ck)
    b = -(2/N)*np.imag(ck)

    PN = a[0]/2 + a[-1]*0.5*(np.cos(N*x/2))

    for k in range(1,int(N/2)):
        PN += a[k]*np.cos(k*x) + b[k]*np.sin(k*x)

    return PN
```

In [3]:
```python
f = [6, 10.242640687119284, 2, -2.585786437626905, 2, 1.757359312880716, -6, -5.41421
N = len(f)
j = np.arange(0,N)
xj = j*(2*np.pi/N)

x = np.linspace(0, 2*np.pi)
y = find_PN(f, xj, x)

plt.figure(figsize=(6,6))
plt.plot(x, y, label = 'spectral approximation');
plt.plot(xj, f, 'o', label = 'data');
plt.grid();
plt.xlabel("x")
plt.ylabel('P(x)');
plt.legend();
```



Question 6

In [4]:
```python
N = len(f)
j = np.arange(0,N)
xj = j*(2*np.pi/N)
f = np.exp(np.sin(xj))

x = np.linspace(0, 2*np.pi)
y = find_PN(f, xj, x)

plt.figure(figsize=(6,6))
plt.plot(x, y, label = 'spectral approximation');
plt.plot(xj, f, 'o', label = 'data');
plt.grid();
plt.xlabel("x")
plt.ylabel('P(x)');
plt.legend();
```



In [5]:
```python
def find_PN_prime(f, xj, x):
    N = len(f)
    ck = np.fft.fft(f)
    a = (2/N)*np.real(ck)
    b = -(2/N)*np.imag(ck)

    PN_prime = a[-1]*0.5*(N/2)*(np.sin(N*x/2))

    for k in range(1,int(N/2)):
        PN_prime += k*a[k]*np.sin(k*x) - k*b[k]*np.cos(k*x)

    return PN_prime
```

In [6]:
```python
N = len(f)
j = np.arange(0,N)
xj = j*(2*np.pi/N)

x = np.linspace(0, 2*np.pi)
y = find_PN_prime(f, xj, x)
fprime_true = -np.cos(x)*np.exp(np.sin(x))

plt.figure(figsize=(6,6))
plt.plot(x, y, label = 'spectral approximation');
plt.plot(x, fprime_true, '--', label = 'actual');
plt.grid();
plt.xlabel("x")
plt.ylabel('P\'(x)');
plt.legend();

print("error (MSE)= ", np.mean((y - fprime_true)**2))
```

```
error (MSE)=  0.00024026733847429038
```