Energy and power of a signal
Consider a continuous time deterministic signal $x(t)$. We are specifically interested in analyzing the characteristics of the signal over the time range $[T_{\text{begin}}$ to $T_{\text{end}}$. To do this, let us defined a “windowed” version of $x(t)$:

$$x_T(t) = \begin{cases} x(t) & \text{for } T_{\text{begin}} \leq t \leq T_{\text{end}} \\ 0 & \text{for all other times} \end{cases} \quad (1)$$

Suppose further that we sample the signal with a sampling frequency $f_s$. We can write $x_T[n] = x[n] = x(t_n)$, where $t_n$ are the sampling times:

$$t_n = T_{\text{begin}} + (n - 1)T_s, \quad n = 1 \ldots N_0 \quad (2)$$

The last time $t_{N_0} = T_{\text{end}}$.

Instantaneous power [W] at each discrete time $t_n$.

$$p[n] = x[n]^2 \quad (3.a)$$

$$p = x.*^2 \quad \text{MATLAB) (3.b)}$$

Average power $P_g [W]$:

$$P_g = \frac{1}{N_0} \sum_n x[n]^2 \quad (4.a)$$

$$P_g = (1/N0).*\text{sum(x.*^2) \quad (MATLAB) (4.b)}$$

Total energy $E_g [J]$:

$$E_g = T_s \sum_n x[n]^2 \quad (5.a)$$

$$E_g = 1/f\text{s*sum(x.*^2) \quad (MATLAB) (5.b)}$$

Spectrum of a signal
The two-sided discrete Fourier transform of a signal can be found in MATLAB can be found with the following command:

$$x = 1/f\text{s*fftshift(fft(x, N)) \quad (MATLAB) (6)}$$

where $N$ is the number of frequency points in the FFT. The discrete Fourier transform $X$ has length $N = \text{length}(X)$, and is defined at discrete frequency points $f_k$:

$$f_k = -\frac{f_s}{2} + (k - 1)\frac{f_s}{N-1}, \quad n = 1 \ldots N \quad (7)$$

Recall that $N_0 = \text{length}(x)$ is the number of discrete time points of the original signal $x[n]$. In general, $N_0 \neq N$, and you are free to choose $N$ as large as you want (so long as your computer can handle it). In fact, you will generally set $N \gg N_0$. A good value for $N$ is $N=2^{15}$. 
**Power spectral density**
In two lines of code you can compute the power spectral density (PSD):

\[
X = \frac{1}{fs} \text{fftshift} (\text{fft}(x,N)) \quad \text{(MATLAB)} \quad (8.a)
\]

\[
\text{PSD} = (\text{abs}(X)^2)/(\text{(N0-1)*Ts}) \quad \text{(MATLAB)} \quad (8.b)
\]

Alternatively, you can also use MATLAB’s built in “periodogram” function:

\[
\text{PSD} = \text{periodogram}(x,[],N,fs,’centered’) \quad \text{(MATLAB)} \quad (9)
\]

**Average Power**
Once you have the PSD, you can compute the average power (\(P_g\)):

\[
P_g = \frac{fs}{N} \times \text{sum} (\text{PSD}) \quad \text{(MATLAB)} \quad (10)
\]

**Autocorrelation**
The autocorrelation function can be calculated with the following MATLAB command.

\[
R_{xx} = \frac{1}{fs} \times \text{xcorr}(x,x) \quad \text{(MATLAB)} \quad (11)
\]

Below are the equations to calculate the signal’s energy and power spectral density using the autocorrelation function:

*Total energy \(E_g\) [J]:*

\[
E_g = \text{max} (R_{xx}) \quad \text{(MATLAB)} \quad (12)
\]

*PSD [W/Hz]:*

\[
\text{PSD} = \frac{1}{(N0-1)*\text{abs(fftshift(fft(Rxx,N)))}} \quad \text{(MATLAB)} \quad (13)
\]

**Noise**
You can create white Gaussian noise in MATLAB with the following commands:

\[
\text{noise_PSD} = \ldots \%\text{choose noise PSD [W/Hz]} \quad \text{(MATLAB)} \quad (14.a)
\]

\[
\text{variance} = \ldots \text{noise_PSD*fs} \quad \text{(MATLAB)} \quad (14.b)
\]

\[
\text{sigma} = \ldots \text{sqrt(variance)} \quad \text{(MATLAB)} \quad (14.c)
\]
In the MATLAB code below, I create a noisy sinusoid and compute/generate the following information:

- Plot histogram of the noise
- Plot of noisy sinusoid in the time domain
- Signal-to-noise ratio (SNR)
- Plots of PSD computing using methods presented above
- Average PSD of noise
- Average value of noise

```matlab
noise = transpose(sigma*randn(N0,1))  \textbf{(MATLAB)} (14.d)
```

```matlab
%Clear variables. clear command window, close all figures:
clc;
clear all;
close all;
%%%Setup and define variables
f0=10; \%frequency of sinusoidal signal (Hz)
fs=100; \%sampling frequency (Hz)
Ts=1/fs; \%sampling period (seconds)
N0=3000; \%number of samples
t=[0:Ts:Tsn*(N0-1)]; \%Sample times
noise_PSD=.5; \%This is the desired noise power spectral density in W/Hz.
variance=noise_PSD*fs; \% Variance = sigma^2
sigma=sqrt(variance);
noise=transpose(sigma*randn(N0,1)); \%create sampled white Gaussian noise.
xsignal=20*sin(2*pi*f0*t); \%create sampled sinusoidal signal
x=xsignal+noise; \%Add signal to noise

figure(1)
histogram(noise,30) \%plot histogram
set(gca,\'FontSize\',14) \%set font size of axis tick labels to 18
xlabel(\'Noise amplitude\',\'fontsize\',14)
ylabel(\'Frequency of occurance\',\'fontsize\',14)
title(\'Simulated histogram of white Gaussian noise\',\'fontsize\',14)
grid on

SNR_try1=snr(xsignal,noise) \%calculate SNR using built in "snr" function.
SNR_try2=10*log10(sum(xsignal.^2)/sum(noise.^2)) \%manually calculate SNR.
%If everything is correct, the two SNR calculations above should agree.

figure(2)
plot(t,x)
set(gca,\'FontSize\',14) \%set font size of axis tick labels to 18
xlabel(\'Time (s)\',\'fontsize\',14)
ylabel(\'Amplitude\',\'fontsize\',14)
title(\'Noisey sinusoid\',\'fontsize\',14)
grid on

%Plot power spectral density (PSD) of noise using three different methods:
```
% Method 1. Calculate PSD from amplitude spectrum
N=2^16; % Number of discrete points in the FFT.
y=fft(x,N)/fs; % fft of noise
z=fftfshift(y); % center noise spectrum
f_vec=[0:1:N-1]*fs/N-fs/2; % designate sample frequencies
amplitude_spectrum=abs(z); % compute two-sided amplitude spectrum
ESD1=amplitude_spectrum.^2; % ESD = |F(w)|^2;
PSD1=ESD1/((N0-1)*Ts); % PSD=ESD/T where T = total time of sample
figure(3)
plot(f_vec,10*log10(PSD1));
xlabel('Frequency [Hz]', 'fontsize',14)
ylabel('dB/Hz', 'fontsize',14)
title('Power spectral density - method 1', 'fontsize',14)
grid on
set(gcf, 'color', 'w'); % set background color from grey (default) to white
axis tight

% calculate average power using PSD calculated from method 1:
Average_power_method_1=sum(PSD1)*fs/N
% Pav=sum(PSD)*delta_f where 
delta_f=fs/N

% Method 2 - Calculate PSD from autocorrelation

time_lag=((length(x)+1):1:(length(x)-1))*Ts;
auto_cor=xcorr(x,x)/fs; % Use xcorr function to find PSD
y=1/fs*fft(auto_cor,N); % fft of auto correlation function
PSD2=abs(1/(N0-1)*fftfshift(fft(auto_cor,N)));
figure(4)
plot(f_vec,10*log10(PSD2)); % use convolution
xlabel('Frequency [Hz]', 'fontsize',14)
ylabel('dB/Hz', 'fontsize',14)
title('Power spectral density - method 2', 'fontsize',14)
grid on
set(gcf, 'color', 'w'); % set background color from grey (default) to white
axis tight
% calculate average power using PSD calculated from method 1:
Average_power_method_2=sum(PSD2)*fs/N
% Pav=sum(PSD)*delta_f where 

% Method 3 - Calculate PSD using built in pwelch function
figure(5)
PSD3=periodogram(x,[],N,fs,'centered');
plot(10*log10(PSD3))
xlabel('Frequency [Hz]', 'fontsize',14)
ylabel('dB/Hz', 'fontsize',14)
title('Power spectral density - method 3', 'fontsize',14)
grid on
set(gcf, 'color', 'w'); % set background color from grey (default) to white
axis tight
Average_power_method_3=sum(PSD3)*fs/N
% Pav=sum(PSD)*delta_f where 

% Calculate mean and average PSD of noise:
PSD_noise=periodogram(noise,[],N,fs,'centered');
Average_noise_PSD=mean(PSD_noise)
Mean_noise=mean(noise)
Results of running MATLAB code:

SNR_try1 =
6.3039

SNR_try2 =
6.3039

Average_power_method_1 =
245.5948

Average_power_method_2 =
245.5948

Average_power_method_3 =
245.5129

Average_noise_PSD =
0.4684

Mean_noise =
0.0983
Simulated histogram of white Gaussian noise

Noisey sinusoid
The power spectral density plots for methods 2 and 3 exactly match that for method 1 (shown above).