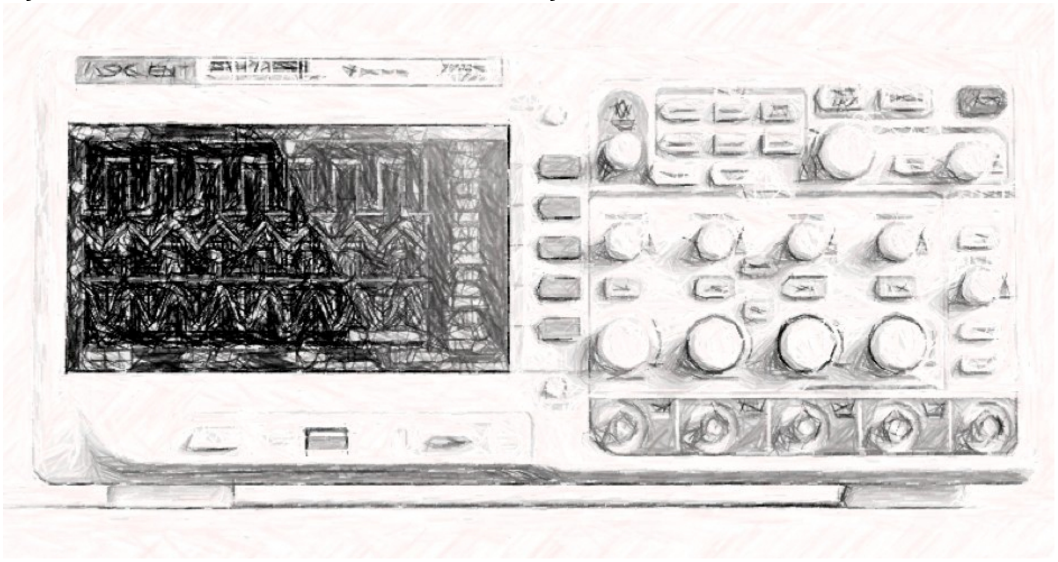
Elementary signal generation with Python

MAY 12, 2018JUNE 26, 2019 THOMAS GAMSJÄGER LEAVE A COMMENT



Being able to simulate your own data is an important prerequisite for testing your algorithms in a reproducible way. Even though there are infinitely many different signals out there, the most fundamental ones are only three: Sine, square and triangle waves. Python makes it an easy task to generate all of them.

First, we have to call in the necessary Python libraries:

```
import numpy as np
from scipy import signal as sg
import matplotlib.pyplot as plt
```

All three waveforms are defined by their frequency (number of oscillations per time unit) and amplitude. (For now, we will leave out the phase parameter.) And in any case, we need a time vector. To create it the np.linspace command renders its useful services. (Just to recap its arguments: start (inclusive), stop (inclusive), number of points.)

```
freq = 2
amp = 2
time = np.linspace(0, 2, 1000)
```

With that, all the ingredients are prepared.

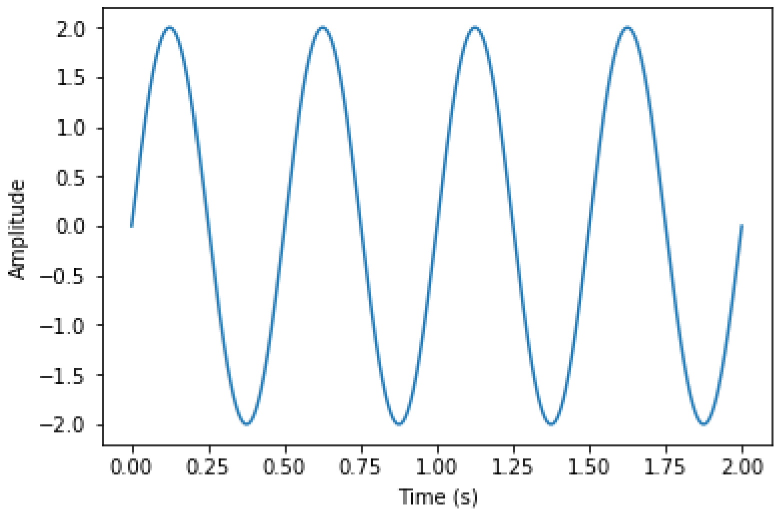
Sine wave

```
signal1 = amp*np.sin(2*np.pi*freq*time)
```

Note that this formula is nothing but the fundamental definition of the sine wave. The term 2*π*freq is also known as the angular frequency ω (lower case omega). The np. prefix tells us that these functions (sin and π) are courteously provided by numpy.

Now for the plot:

```
plt.plot(time, signal1)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()
```



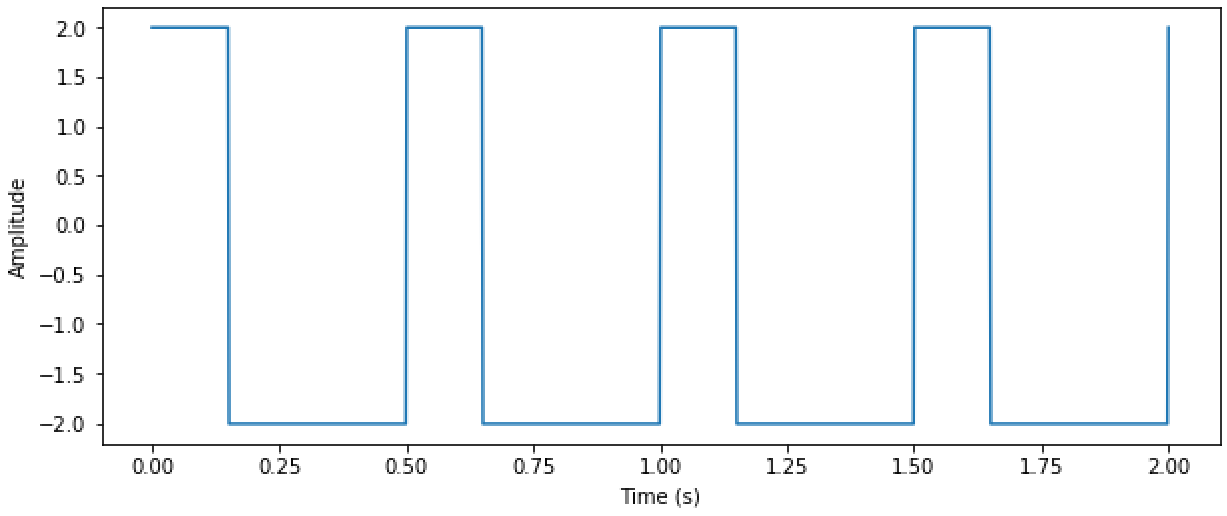
Square wave

```
signal2 = amp*sg.square(2*np.pi*freq*time, duty=0.3)
```

Here we use the ‘square’ tool from the scipy/signal library. The optional argument ‘duty’ defines which fraction of the whole duty cycle the signal will be in its ‘high’ state.

The plotting commands are pretty much the same as above so we can leave them out here with one exception: I would like to introduce ‘figsize’, with which it is possible to define, you guessed it, the size of the figure.

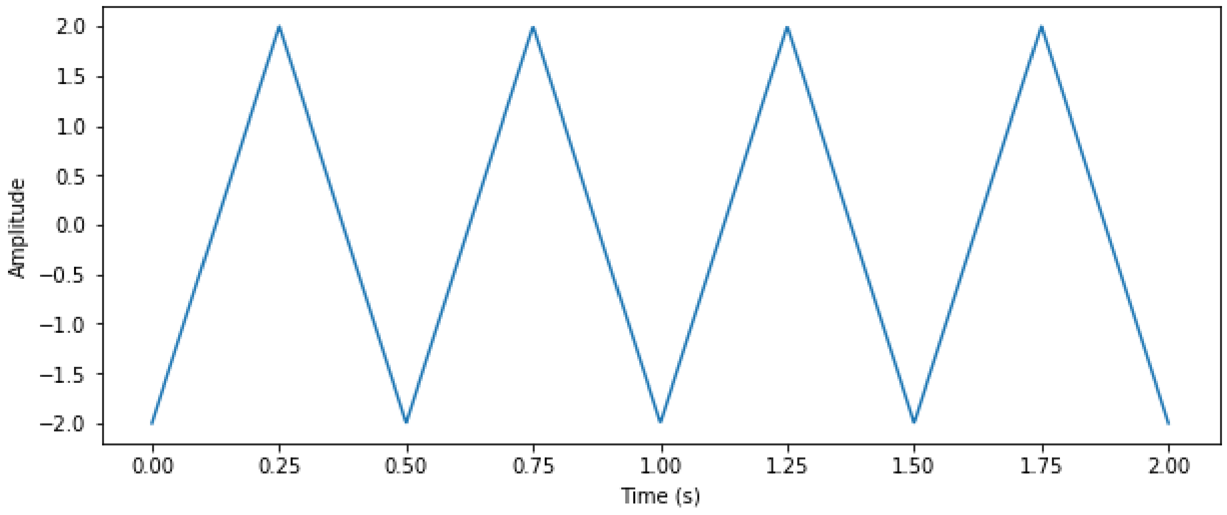
```
plt.figure(figsize=(10,4))
```



Triangle wave

```
signal3 = amp*sg.sawtooth(2*np.pi*freq*time, width=0.5)
```

The triangle wave is a special case of the sawtooth wave where the signal takes exactly the same amount of time to rise as it takes to fall. The optional argument width=0.5 does just that.


Amplitude
Time (s)