In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Exercice Numpy

1. Array creation

- Create a 1D array with values from 0 to 10 and in steps of 0.1. Check the shape of the array:

In [145]:
```
xarray = np.arange(0,10,0.1)
xarray.shape
```

Out[145]:
```
(100,)
```

1.2. Create an array of normally distributed numbers with mean $\mu = 0$ and standard deviation $\sigma = 0.5$. It should have 20 rows and as many columns as there are elements in `xarray` . Call it `normal_array` :

In [146]:
```
normal_array = np.random.normal(0,0.5,(20, xarray.shape[0]))
```

- Check the type of `normal_array` :

In [147]:
```
normal_array.dtype
```

Out[147]:
```
dtype('float64')
```

2. Array mathematics

- Using `xarray` as x-variable, create a new array `yarray` as y-variable using the function $y = 10 * cos(x) * e^{-0.1x}$:

In [148]:
```
yarray = 5*np.cos(xarray)*np.exp(-0.1*xarray)
```

- 2.2 Create `array_abs` by taking the absolute value of `array_mul` :

In [149]:
```
array_abs = np.abs(yarray)
```

- 2.2 Create a boolan array (logical array) where all positions > 0.3 in `array_abs` are `True` and the others `False`

In [165]:
```
array_bool = array_abs > 0.3
```

- 2.3 Create a standard deviation projection along the second dimension (columns) of `array_abs` . Check that the dimensions are the ones you expected. Also are the values around the value you expect?

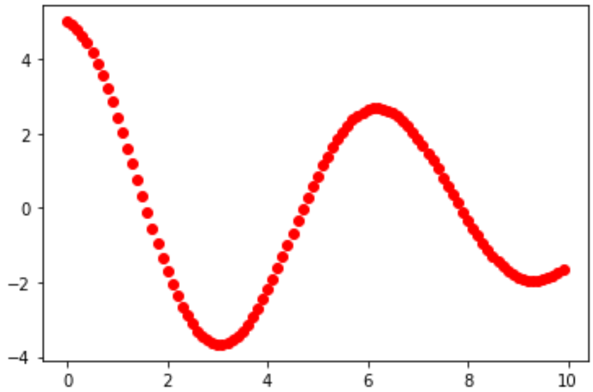
In [167]:
```
array_min = normal_array.std(axis = 1)
array_min.shape
```
Out[167]:
```
(20,)
```

In [168]:
```
array_min
```
Out[168]:
```
array([0.54167658, 0.51651789, 0.4832876 , 0.54537271, 0.50834276,
       0.47623427, 0.44677832, 0.47841273, 0.50255308, 0.50656681,
       0.47822978, 0.52051232, 0.55511136, 0.46977863, 0.57914545,
       0.47393849, 0.52705922, 0.43786828, 0.55795931, 0.45476456])
```

3. Plotting

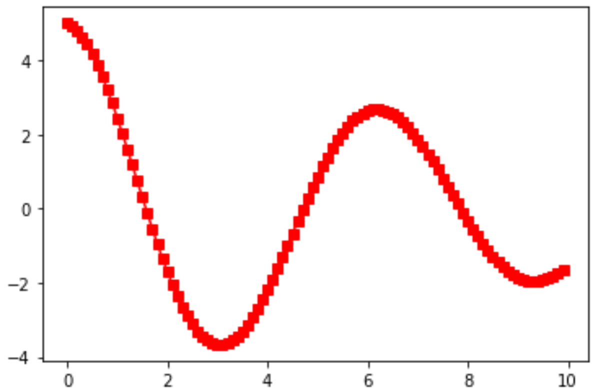
- Use a line plot to plot `yarray` vs `xarray` :

In [172]:
```
plt.plot(xarray, yarray,'ro')
```
Out[172]:
```
[<matplotlib.lines.Line2D at 0x11fb2b9d0>]
```



- Try to change the color of the plot to red and to have markers on top of the line as squares:

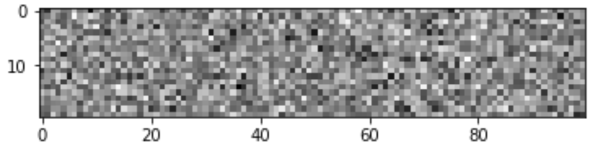
In [174]:
```
plt.plot(xarray, yarray, '-sr')
```
Out[174]:
```
[<matplotlib.lines.Line2D at 0x11f806070>]
```



- Plot the `normal_array` as an imagage and change the colormap to 'gray':

In [175]:
```
plt.imshow(normal_array, cmap = 'gray')
```

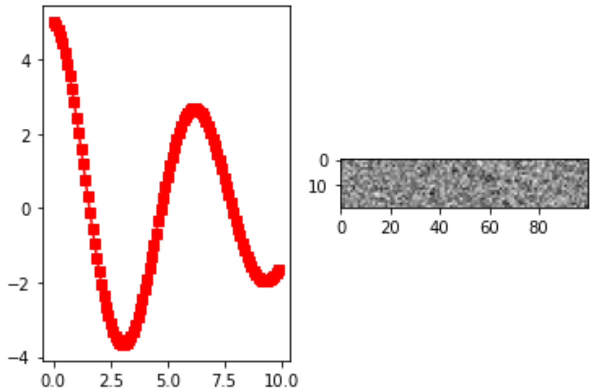
Out[175]: <matplotlib.image.AxesImage at 0x11fd9dfd0>



- Assemble the two above plots in a figure with one row and two columns grid:

In [176]:
```
fig, ax = plt.subplots(1,2)
ax[0].plot(xarray, yarray, '-sr')
ax[1].imshow(normal_array, cmap = 'gray')
```

Out[176]: <matplotlib.image.AxesImage at 0x11fd9a340>



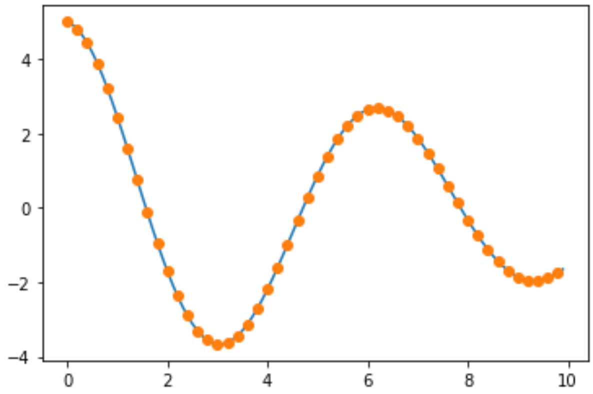
4. Indexing

- Create new arrays where you select every second element from xarray and yarray. Plot them on top of `xarray` and `yarray` .

In [179]:
```
xarray2 = xarray[::2]
yarray2 = yarray[::2]

plt.plot(xarray, yarray)
plt.plot(xarray2, yarray2,'o')
```

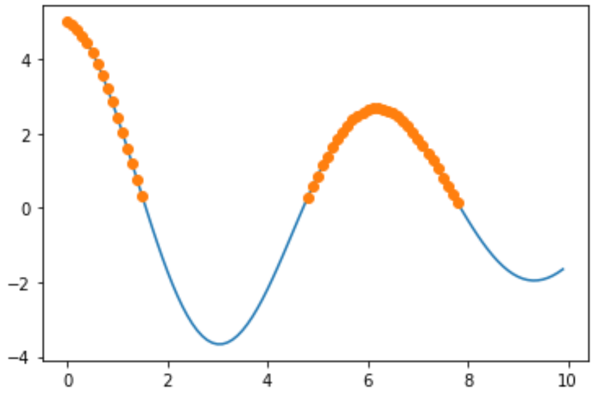
Out[179]: [<matplotlib.lines.Line2D at 0x12045fbb0>]



- Select all values of `yarray` that are larger than 0. Plot those on top of the regular `xarray` and `yarray` plot.

In [181]:
```
plt.plot(xarray, yarray)
plt.plot(xarray[yarray>0], yarray[yarray>0],'o')
```

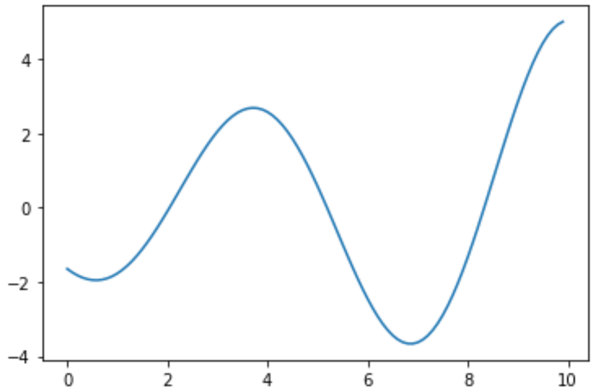
Out[181]: [<matplotlib.lines.Line2D at 0x1205f0880>]



- Flip the order of `xarray` use it to plot `yarray` :

In [185]:
```
flipped_array = np.flipud(xarray)
plt.plot(flipped_array, yarray)
```

Out[185]: [<matplotlib.lines.Line2D at 0x120848dc0>]

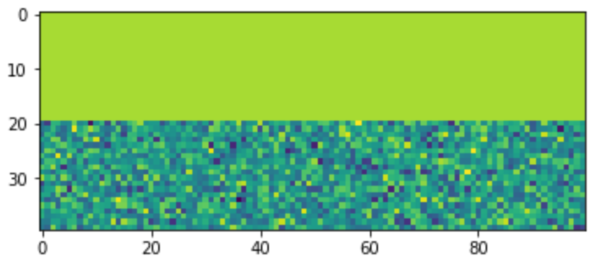


5. Combining arrays

- Create an array filled with ones with the same shape as `normal_array` . Concatenate it to `normal_array` along the first dimensions and plot the result:

In [189]:
```
ones_array = np.ones(normal_array.shape)
concatenated = np.concatenate([ones_array, normal_array])

plt.imshow(concatenated);
```



- `yarray` represents a signal. Each line of `normal_array` represents a possible random noise for that signal. Using broadcasting, try to create an array of noisy versions of `yarray` using `normal_array` . Finally, plot it:

The last dimensions of both arrays are matching. We can therefore simply added the two arrays, and `yarray` will simply be "replicated" as many times as needed:

In [194]:
```
yarray_noise = yarray + normal_array
```

In [196]:
```
plt.imshow(yarray_noise)
```

Out[196]: <matplotlib.image.AxesImage at 0x11b249b80>