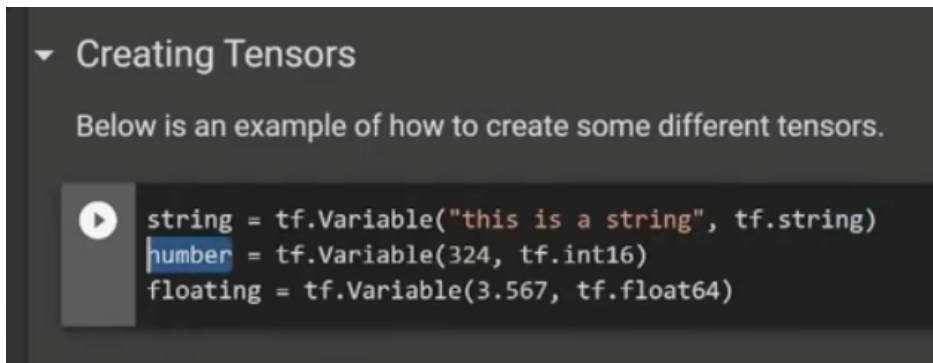


- **What are tensors**

- -> vectors
- -> a tensor is a higher dimension vector or matrix <- it's the class which overarches over both of them
- -> datapoints
- -> i.e they can be negative (not just scalars)
- -> they can be negative and n dimensional, where n can be > 3
- -> these are the objects which are passed around the layers of the neural networks
  - -> to transform the data across their different layers and give us the output of the model
- -> the datatype is the type of information which the tensor holds -> strings, integers
- -> there are also different shapes -> aka the dimensions of the tensors

- **An example used to create tensors in Python**



```
▼ Creating Tensors

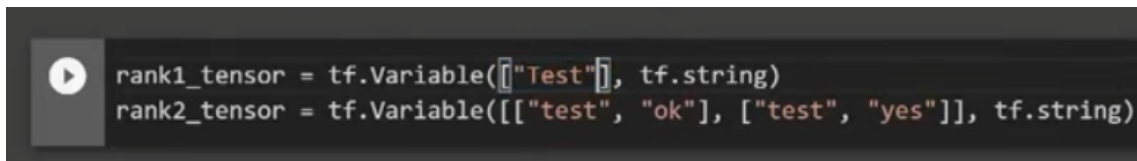
Below is an example of how to create some different tensors.

string = tf.Variable("this is a string", tf.string)
number = tf.Variable(324, tf.int16)
floating = tf.Variable(3.567, tf.float64)
```

- -> tf.Variable -> then the value and the datatype of the tensor
  - the datatype goes second
- -> the shapes of these tensors are 1
- -> in other words vectors

- **Rank / degree of the tensors <- the number of dimensions**

- -> a number has zero dimensions -> it's rank 0
  - scalars



```
rank1_tensor = tf.Variable(["Test"], tf.string)
rank2_tensor = tf.Variable(["test", "ok"], ["test", "yes"], tf.string)
```

- compared to this example where he is creating tensors which have a higher rank
  - -> in this case it's a list which is nested include another list
  - -> there are two layers of embedded arrays in this example - so it's a rank 2 tensor
- **to determine the rank <- number of dimensions (zero indexed) which a tensor covers**
  - -> tf.rank(name\_of\_tensor)
  - -> then it tells you the number of dimensions (zero indexed) which that tensor lies across
- **to determine the shape <- rows and columns which the tensor covers**
  - -> name\_of\_tensor.shape
  - -> then it outputs the row and columns of the tensor which was input into it
  - -> that method has no arguments
  - -> if you change the values and re-run the code then the features update
- **to change the shape**
  - -> the same number of elements but in a different number of rows and columns
  - -> flattening a tensor means taking it and removing it's shape -> so you can then take the same data and set the rows and columns you want it to have
  - -> tensor\_name = tf.reshape(tensor\_name\_two, [row, col]) <- the code to do this
  - -> it's possible to take the data and reshape it into another different shape
    - -> to change the number of rows and columns - the same data is organised in

- **Types of tensors**

- -> Variables
- -> Constants
- -> Placeholder
- -> sparseTensor

- -> all of their values are **immutable** (unchangeable) -> compared to a variable tensor
  - -> the value of certain tensors can't change once it's been defined
- **Sessions <- getting the value of tensors while calculations are being ran**
  - -> you run a session to get the values of tensor objects -> e.g if you're half way through a calculation with it
  - -> the simplest way of doing this is

```
[ ] with tf.Session() as sess: # creates a session using the default graph
    tensor.eval() # tensor will of course be the name of your tensor
```

- -> you iterate through the tensor and evaluate each of the different values which it stores
- -> it's like doing `print` when iterating through a numpy array - to get the different values it stores
- -> there is a default graph for a tensor -> these are the values this method outputs
- -> this is from the tensor flow website / guide
- **Examples of reshaping**
  - -> he's imported tensor flow
  - -> then made a tensor called `t` -> `tf.ones()` <- a vector of ones and the argument is the shape of it
    - -> he's done `.zeros`
  - -> then printed that matrix and reshaped it
  - -> to reshape it
    - -> he's done `tf.reshape(t,[625])`
    - -> it's reshaped the matrix
    - -> if the argument is e.g `[125, -1]` <- then it's inferred what the shape of that tensor needs to be
    - -> and sessions are where he's printed the value of that variable

```
%tensorflow_version 2.x
import tensorflow as tf
print(tf.version)

t = tf.zeros([5,5,5,5])
|
t = tf.reshape(t, [125, -1])
print(t)
```