- -> he imports tensor flow as tfp <- this is a separate library which deals with probability

```
import tensorflow_probability as tfp  # We are using a different module from tensorflow this time
import tensorflow as tf
```

- ***The weather model using a Hidden Markov model -> probability distributions***

1. Cold days are encoded by a 0 and hot days are encoded by a 1.
2. The first day in our sequence has an 80% chance of being cold.
3. A cold day has a 30% chance of being followed by a hot day.
4. A hot day has a 20% chance of being followed by a cold day.
5. On each day the temperature is normally distributed with mean and standard deviation 0 and 5 on a cold day and mean and standard deviation 15 and 10 on a hot day.

  - -> these (above) are the rules of the model - in this example
  - -> if something is ± ..., then the standard deviation is almost like the average uncertainty that a datapoint will have in that probability distribution
  - -> he loads the tensor flow probability distributions model

```
tfd = tfp.distributions  # making a shortcut for later on
initial_distribution = tfd.Categorical(probs=[0.8, 0.2])  # Refer to point 2 above
transition_distribution = tfd.Categorical(probs=[[0.7, 0.3],
                                                 [0.2, 0.8]])  # refer to points 3 and 4 above
observation_distribution = tfd.Normal(loc=[0., 15.], scale=[5., 10.])  # refer to point 5 above

# the loc argument represents the mean and the scale is the standard devitation
```

    - -> the initial distribution is 80% cold and 20% otherwise
    - -> he's entering the values from the question into the model
    - -> this entire cell is getting the word equations in the question into code in the model
    - -> then the observation distribution
      - it's a normal distribution
      - then loc is the average / mean distrubution
      - -> he's creating different probability distributions for the model, based off of the parameters in the question
  - -> then he creates the model

```
model = tfd.HiddenMarkovModel(
    initial_distribution=initial_distribution,
    transition_distribution=transition_distribution,
    observation_distribution=observation_distribution,
    num_steps=7)
```

    - -> the inputs to this model are the probability distributions which were defined based off of the information in the question (in the previous cell)
    - -> steps is the number of days we want to predict for into the future (the number of times we are running the model) -> and then based off of those we can predict what the average temperature will be

    - -> these commends are to ensure that the correct version of tensor flow is installed
      - -> he then re-runs the model

```
[2] %tensorflow_version 2.x  # this line is not required unless you are in a notebook

`%tensorflow_version` only switches the major version: `1.x` or `2.x`.
You set: `2.x  # this line is not required unless you are in a notebook`. This will be interpreted as:


TensorFlow 2.x selected.
```

Due to a version mismatch with tensorflow v2 and tensorflow_probability we need to install the most recent version o
(see below).

```
!pip install tensorflow_probability==0.8.0rc0 --user --upgrade
```

```
[ ] import tensorflow_probability as tfp  # We are using a different module from tensorflow this time
    import tensorflow as tf
```

- -> to run the model and see the output

```
mean = model.mean()

# due to the way TensorFlow works on a lower level we need to evaluate part of the graph
# from within a session to see the value of this tensor

# in the new version of tensorflow we need to use tf.compat.v1.Session() rather than just tf.Session()
with tf.compat.v1.Session() as sess:
  print(mean.numpy())
```

```
[2.9999998 5.9999995 7.4999995 8.25      8.625001  8.812501  8.90625  ]
```

  - -> model.mean <- this returns the mean of the data based off of the future predicted data for the weather
  - -> session as sess -> running the model and predicting the mean as it's ran
  - -> it's returning the expected temperatures on each day
- -> he then makes changes to the initial probability distributions to see how the outcomes change
  - -> what the model predicts
  - -> in this case the starting temperature is the same but the outcomes are different -> he's changed the probability that given where we are, the next state will be this
  - **-> hidden Markov models are like quantum mechanics with the probability distributions**
    - **-> given where we are, what is the next most likely state which the wave function could collapse onto?**
    - **-> and the different states are hidden (hidden Markov model, because this is the probabilistic one)**
    - **-> done with tensorflow probability in this case**

- ***The content in this module covered***
  - -> implementing machine learning algorithms
  - -> testing vs training models with data
  - -> linear regression
  - -> classification
  - -> clustering -> K means
  - -> the next module is neural networks then chatbots with recurrent neural networks