✓ **Congratulations! You passed!**

TO PASS 80% or higher

Keep Learning

GRADE
100%

Custom Layers

LATEST SUBMISSION GRADE

100%

1. Lambda layer allows to execute an arbitrary function only within a Sequential API model. **1 / 1 point**

- True
- [✓ False]

✓ **Correct**
Correct!

2. Which one of the following is the correct syntax for mapping an increment of 2 to the value of "x" using a Lambda layer? (tf = Tensorflow) **1 / 1 point**

- [✓ tf.keras.layers.Lambda(lambda x: tf.math.add(x, 2.0))]

- True
- [✓ False]

✓ **Correct**
Correct!

4. A *Layer* is defined by having "States" and "Computation". Consider the following code and check all that are true: **1 / 1 point**

```
    self.b = tf.Variable(name="bias",
                         initial_value=b_init(shape=(self.units,), dtype='float32'),
                                                            trainable=False)

  def call(self, inputs):
    return tf.matmul(inputs, self.w) + self.b
```

- ☐ def call(self, inputs): performs the computation and is called when the Class is instantiated.
- ☑ You use def build(self, input_shape): to create the state of the layers and specify local input states.

```
class SimpleDense(Layer):

  def __init__(self, units=32):
    super(SimpleDense, self).__init__()
    self.units = units

  def build(self, input_shape):
    w_init = tf.random_normal_initializer()
    self.w = tf.Variable(name="kernel",
                         initial_value=w_init(shape=(input_shape[-1], self.units),
                                              dtype='float32'), trainable=True)
```

implementation:

- def build(self, input_shape):

  .

  .

  self.activation = tf.keras.activations.get(activation)

  def call(self, inputs):

  return self.activation(tf.matmul(inputs, self.w) + self.b)

- def __init__(self, units=32):

  .

  .

  self.activation = tf.keras.activations.get(activation)

  def call(self, inputs):

  return self.activation(tf.matmul(inputs, self.w) + self.b)

- [✓ def __init__(self, units=32, activation=None):]

  .

  .

  self.activation = tf.keras.activations.get(activation)

  def call(self, inputs):

  return self.activation(tf.matmul(inputs, self.w) + self.b)

✓ **Correct**
Correct!