|  |
| --- |
|  |

|  |  |
| --- | --- |
|  | |
| **Solution** :    <https://docs.pennylane.ai/en/stable/code/api/pennylane.BasisStatePreparation.html>  <https://numpy.org/doc/stable/reference/generated/numpy.binary_repr.html>  num\_wires = 3  dev = qml.device("default.qubit", wires=num\_wires)  @qml.qnode(dev)  def make\_basis\_state(basis\_id):  """Produce the 3-qubit basis state corresponding to |basis\_id>.  Note that the system starts in |000>.  Args:  basis\_id (int): An integer value identifying the basis state to construct.  Returns:  np.array[complex]: The computational basis state |basis\_id>.  """  ##################  # YOUR CODE HERE #  bits = [int(x) for x in np.binary\_repr(basis\_id, width=num\_wires)]  qml.BasisStatePreparation(bits, wires=range(num\_wires))  ##################  # CREATE THE BASIS STATE  return qml.state()  basis\_id = 3  print(f"Output state = {make\_basis\_state(basis\_id)}") | |
| **Qiskit Program**:    import numpy as np  import random  import pennylane as qml  import matplotlib.pyplot as plt  num\_wires = 3  dev = qml.device("default.qubit", wires=num\_wires)  @qml.qnode(dev)  def make\_basis\_state(basis\_id):  """Produce the 3-qubit basis state corresponding to |basis\_id>.  Note that the system starts in |000>.  Args:  basis\_id (int): An integer value identifying the basis state to construct.  Returns:  np.array[complex]: The computational basis state |basis\_id>.  """  ##################  # YOUR CODE HERE #  # Prepare the basis state |basis\_id>  #Option 1:  #bits = [int(x) for x in np.binary\_repr(basis\_id, width=num\_wires)]  #qml.BasisStatePreparation(bits, wires=[0, 1, 2])  #Option 2:  bits = [int(x) for x in np.binary\_repr(basis\_id, width=num\_wires)]  qml.BasisStatePreparation(bits, wires=range(num\_wires))  ##################  # CREATE THE BASIS STATE  return qml.state()  basis\_id = 3  print(f"Output state = {make\_basis\_state(basis\_id)}")  **O/P:** |

|  |
| --- |
|  |
| **Solution:**  # Creates a device with \*two\* qubits  dev = qml.device("default.qubit", wires=2)  @qml.qnode(dev)  def two\_qubit\_circuit():  ##################  # YOUR CODE HERE #    # PREPARE |+>  qml.Hadamard(wires=0)    # PREPARE |1>  qml.X(wires=1)  ##################  # RETURN TWO EXPECTATION VALUES, Y ON FIRST QUBIT, Z ON SECOND QUBIT  return qml.expval(qml.PauliY(0)), qml.expval(qml.PauliZ(1))  print(two\_qubit\_circuit()) |
| **Qiskit Program**:  import numpy as np  import random  import pennylane as qml  import matplotlib.pyplot as plt  # Creates a device with \*two\* qubits  dev = qml.device("default.qubit", wires=2)  @qml.qnode(dev)  def two\_qubit\_circuit():  ##################  # YOUR CODE HERE #    # PREPARE |+>  qml.Hadamard(wires=0)    # PREPARE |1>  qml.X(wires=1)  ##################  # RETURN TWO EXPECTATION VALUES, Y ON FIRST QUBIT, Z ON SECOND QUBIT  #return qml.probs(wires=[0, 1])  return qml.expval(qml.PauliY(0)), qml.expval(qml.PauliZ(1))  print(two\_qubit\_circuit())  **O/P**: |

|  |
| --- |
|  |
| **Solution:**  dev = qml.device("default.qubit", wires=2)  @qml.qnode(dev)  def create\_one\_minus():  ##################  # YOUR CODE HERE #  ##################  # PREPARE |1>|->  # PREPARE |1>  qml.X(wires=0)  # PREPARE |->  qml.X(wires=1)  qml.Hadamard(wires=1)    # RETURN A SINGLE EXPECTATION VALUE Z \otimes X  op = qml.PauliZ(0) @ qml.PauliX(1)  return qml.expval(op)  print(create\_one\_minus()) |
| **Qiskit Program**:  import numpy as np  import random  import pennylane as qml  import matplotlib.pyplot as plt  # Creates a device with \*two\* qubits  dev = qml.device("default.qubit", wires=2)  @qml.qnode(dev)  def create\_one\_minus():  ##################  # YOUR CODE HERE #  ##################  # PREPARE |1>|->  # PREPARE |1>  qml.X(wires=0)  # PREPARE |->  qml.X(wires=1)  qml.Hadamard(wires=1)    # RETURN A SINGLE EXPECTATION VALUE Z \otimes X  op = qml.PauliZ(0) @ qml.PauliX(1)  return qml.expval(op)  print(create\_one\_minus())  **O/P**: |

|  |
| --- |
| **Refer : https://discuss.pennylane.ai/t/any-thought-on-i11-4/1510/4** |
| **Solution:** |
| **Qiskit Program**: |