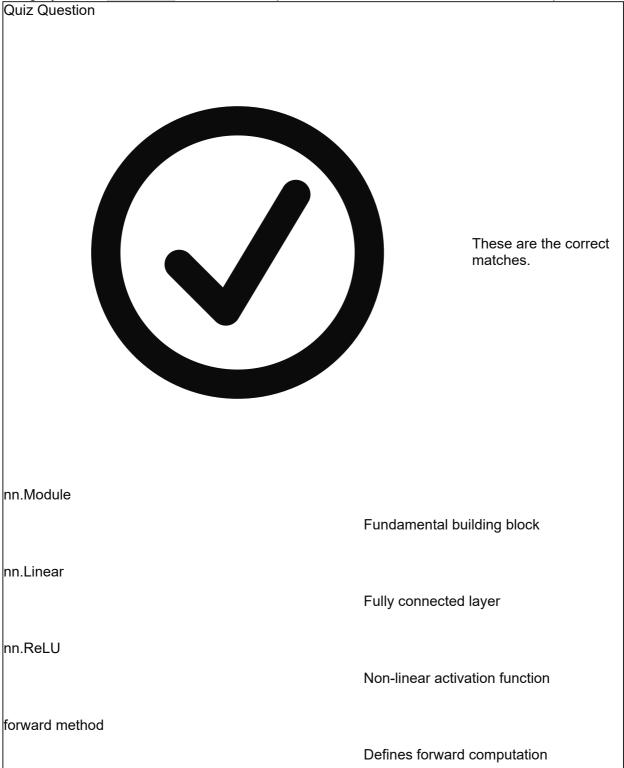
## **Show TranscriptSummarize Video**

PyTorch offers powerful features to create and interlink neural networks, which are key elements in understanding modern artificial intelligence. We explored creating a multi-layer perceptron using PyTorch's nn.Module class and then passed a tensor into it and received the output.



Code Example

```
import torch.nn as nn
class MLP(nn.Module):
    def __init__(self, input_size):
        super(MLP, self).__init__()
        self.hidden_layer = nn.Linear(input_size, 64)
        self.output_layer = nn.Linear(64, 2)
        self.activation = nn.ReLU()
    def forward(self, x):
        x = self.activation(self.hidden_layer(x))
        return self.output layer(x)
model = MLP(input_size=10)
print(model)
# MLP(
    (hidden_layer): Linear(in_features=10, out_features=64, bias=True)
    (output_layer): Linear(in_features=64, out_features=2, bias=True)
    (activation): ReLU()
# )
model.forward(torch.rand(10))
# tensor([0.2294, 0.2650], grad_fn=<AddBackward0>)
```

Resources

**PyTorch nn tutorial** 

**PyTorch nn documentation** 

torch.nn.Module documentation

torch.nn.Linear documentation

torch.nn.ReLU documentation