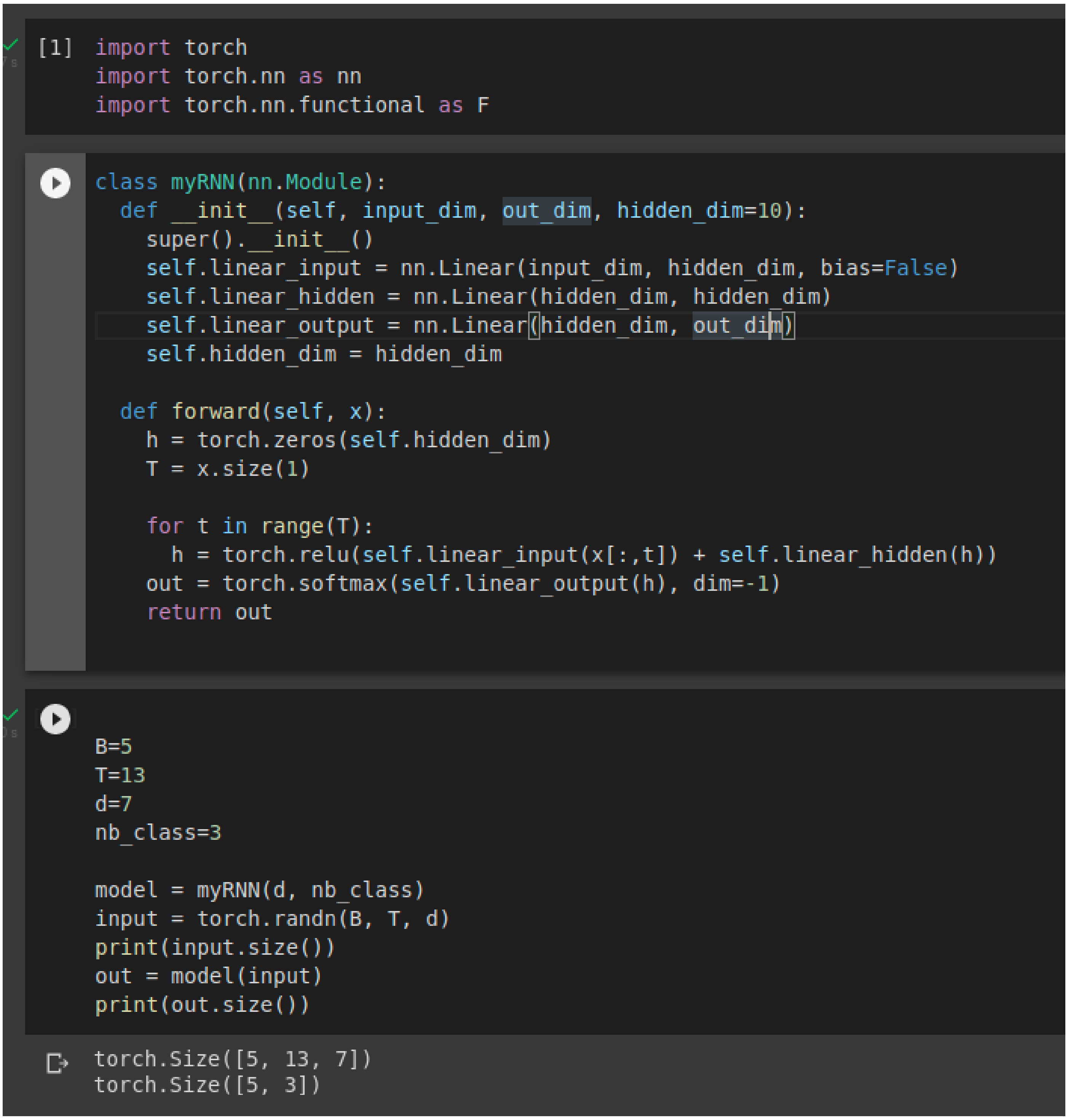

```
[1] import torch
    import torch.nn as nn
    import torch.nn.functional as F
```

```
class myRNN(nn.Module):
  def __init__(self, input_dim, out_dim, hidden_dim=10):
    super().__init__()
    self.linear_input = nn.Linear(input_dim, hidden_dim, bias=False)
    self.linear_hidden = nn.Linear(hidden_dim, hidden_dim)
    self.linear_output = nn.Linear(hidden_dim, out_dim)
    self.hidden_dim = hidden_dim

  def forward(self, x):
    h = torch.zeros(self.hidden_dim)
    T = x.size(1)

    for t in range(T):
      h = torch.relu(self.linear_input(x[:,t]) + self.linear_hidden(h))
    out = torch.softmax(self.linear_output(h), dim=-1)
    return out
```

```

B=5
T=13
d=7
nb_class=3

model = myRNN(d, nb_class)
input = torch.randn(B, T, d)
print(input.size())
out = model(input)
print(out.size())
```

```
torch.Size([5, 13, 7])
torch.Size([5, 3])
```