In [27]:
```
#사용한 모듈
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout, Flatten,
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers, initializers, regularizers, metr
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
from torchvision import datasets, models, transforms
import time
import os
```

In [34]:
```
#gpu 사용을 위한 구문
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cpu
```

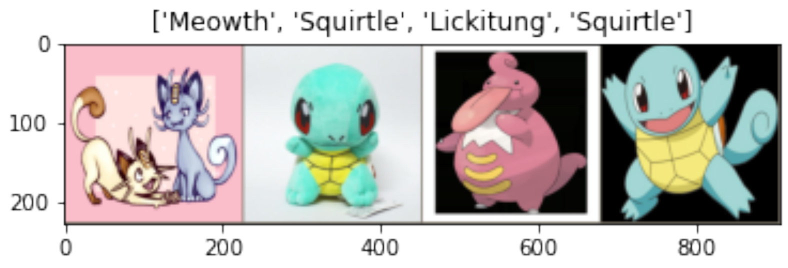
In [29]:
```
# 데이터셋을 불러올 때 사용할 변형(transformation) 객체 정의
transforms_train = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(), # 데이터 증진(augmentation),랜덤 수평 두
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    # 정규화(normalization), scaling(scale 조절), centering(중심 지정)
])

transforms_test = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

data_dir = './'
train_datasets = datasets.ImageFolder(os.path.join(data_dir, 'train'), tr
test_datasets = datasets.ImageFolder(os.path.join(data_dir, 'test'), tran

train_dataloader = torch.utils.data.DataLoader(train_datasets, batch_size
test_dataloader = torch.utils.data.DataLoader(test_datasets, batch_size=4

print('학습 데이터셋 크기:', len(train_datasets))
print('테스트 데이터셋 크기:', len(test_datasets))

class_names = train_datasets.classes
print('클래스:', class_names)
```

```
학습 데이터셋 크기: 811
테스트 데이터셋 크기: 100
클래스: ['Arbok', 'Eevee', 'Koffing', 'Lickitung', 'Meowth', 'Psyduck', 'S
norlax', 'Squirtle', 'Weepinbell', 'Zubat']
```

In [30]:

```python
#학습데이터가 잘 들어 갔는지 확인하기 위해 실행

def imshow(input, title):
    # torch.Tensor를 numpy 객체로 변환
    input = input.numpy().transpose((1, 2, 0))
    # 이미지 정규화 해제하기
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    input = std * input + mean
    input = np.clip(input, 0, 1)
    # 이미지 출력
    plt.imshow(input)
    plt.title(title)
    plt.show()


# 학습 데이터를 배치 단위로 불러오기
iterator = iter(train_dataloader)

# 현재 배치를 이용해 격자 형태의 이미지를 만들어 시각화
inputs, classes = next(iterator)
out = torchvision.utils.make_grid(inputs)
imshow(out, title=[class_names[x] for x in classes])
```

```
Traceback (most recent call last):
  File "/Users/jhs/miniforge3/lib/python3.9/multiprocessing/queues.py", l
ine 251, in _feed
    send_bytes(obj)
  File "/Users/jhs/miniforge3/lib/python3.9/multiprocessing/connection.py
", line 205, in send_bytes
    self._send_bytes(m[offset:offset + size])
  File "/Users/jhs/miniforge3/lib/python3.9/multiprocessing/connection.py
", line 416, in _send_bytes
    self._send(header + buf)
  File "/Users/jhs/miniforge3/lib/python3.9/multiprocessing/connection.py
", line 373, in _send
    n = write(self._handle, buf)
BrokenPipeError: [Errno 32] Broken pipe
```



In [31]:
```
#모델 초기화

model = models.resnet34(pretrained=True)
#vgg19개념에 +전차
num_features = model.fc.in_features

model.fc = nn.Linear(num_features, 10)
# 전이 학습(transfer learning): 모델의 출력 뉴런 수를 10개로 설정하여 마지막 레이어 다시

model = model.to(device)

criterion = nn.CrossEntropyLoss()
#다중분류를 위한 대표적인 손실함수

optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

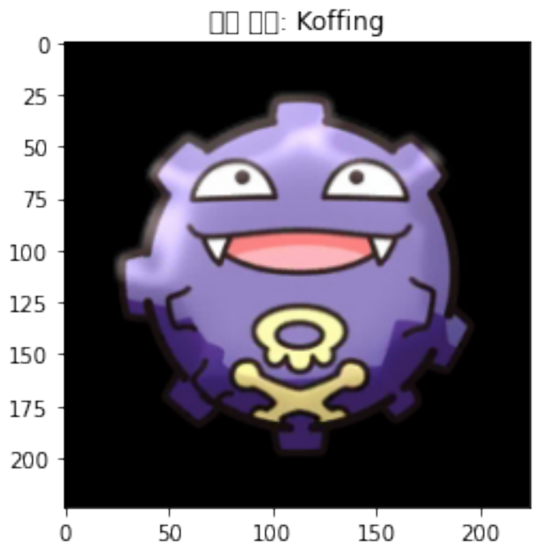
In [32]:
```
#epochs수를 많이 하지 않아도 충분한 정확도가 나오기에 30만 진행

num_epochs = 30
model.train()
start_time = time.time()

# 전체 반복(epoch) 수 만큼 반복하며
for epoch in range(num_epochs):
    running_loss = 0.
    running_corrects = 0

    # 배치 단위로 학습 데이터 불러오기
    for inputs, labels in train_dataloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        # 모델에 입력(forward)하고 결과 계산
        #gradient를 0으로 초기화, torch는 미분값들이 누적되는 특징
        optimizer.zero_grad()
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

        # 역전파를 통해 기울기(gradient) 계산 및 학습 진행
        loss.backward()
        optimizer.step()

        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)

    epoch_loss = running_loss / len(train_datasets)
    epoch_acc = running_corrects / len(train_datasets) * 100.

    # 학습 과정 중에 결과 출력
    print('#{} Loss: {:.4f} Acc: {:.4f}% Time: {:.4f}s'.format(epoch, epo
#0 Loss: 1.0378 Acc: 67.4476% Time: 69.5711s
#1 Loss: 0.3461 Acc: 89.8890% Time: 141.1523s
#2 Loss: 0.2135 Acc: 93.7115% Time: 212.8015s
#3 Loss: 0.1688 Acc: 95.4377% Time: 284.7899s
#4 Loss: 0.1729 Acc: 94.8212% Time: 356.7104s
#5 Loss: 0.1467 Acc: 95.8076% Time: 427.9811s
#6 Loss: 0.0824 Acc: 97.7805% Time: 499.9115s
#7 Loss: 0.0945 Acc: 96.4242% Time: 571.9059s
#8 Loss: 0.0689 Acc: 97.7805% Time: 643.5520s
#9 Loss: 0.0755 Acc: 97.4106% Time: 715.7191s
#10 Loss: 0.0332 Acc: 99.3835% Time: 787.9141s
#11 Loss: 0.0360 Acc: 99.1369% Time: 861.9886s
#12 Loss: 0.0353 Acc: 99.1369% Time: 934.7454s
#13 Loss: 0.0478 Acc: 98.7670% Time: 1008.5240s
#14 Loss: 0.0227 Acc: 99.5068% Time: 1082.2638s
#15 Loss: 0.0370 Acc: 99.2602% Time: 1155.1793s
#16 Loss: 0.0542 Acc: 98.6436% Time: 1228.8428s
#17 Loss: 0.0428 Acc: 98.8903% Time: 1303.4138s
#18 Loss: 0.0361 Acc: 99.2602% Time: 1377.3434s
#19 Loss: 0.0204 Acc: 99.6301% Time: 1452.8820s
#20 Loss: 0.0663 Acc: 98.1504% Time: 1531.4197s
#21 Loss: 0.0579 Acc: 98.3970% Time: 1604.1156s
#22 Loss: 0.0394 Acc: 98.5203% Time: 1678.9360s
#23 Loss: 0.0134 Acc: 99.7534% Time: 1753.5021s
#24 Loss: 0.0481 Acc: 98.7670% Time: 1827.5789s
#25 Loss: 0.0430 Acc: 99.0136% Time: 1902.5643s
#26 Loss: 0.0334 Acc: 99.3835% Time: 1976.5819s
#27 Loss: 0.0289 Acc: 99.2602% Time: 2051.8838s
#28 Loss: 0.0172 Acc: 99.7534% Time: 2126.2446s
#29 Loss: 0.0491 Acc: 98.5203% Time: 2198.6581s
```

In [33]:
```
model.eval()
start_time = time.time()

with torch.no_grad():
    running_loss = 0.
    running_corrects = 0

    for inputs, labels in test_dataloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)

        # 한 배치의 첫 번째 이미지에 대하여 결과 시각화
        print(f'[예측 결과: {class_names[preds[0]]}] (실제 정답: {class_names[
        imshow(inputs.cpu().data[0], title='예측 결과: ' + class_names[pred

    epoch_loss = running_loss / len(test_datasets)
    epoch_acc = running_corrects / len(test_datasets) * 100.
    print('[Test Phase] Loss: {:.4f} Acc: {:.4f}% Time: {:.4f}s'.
          format(epoch_loss, epoch_acc, time.time() - start_time))
```

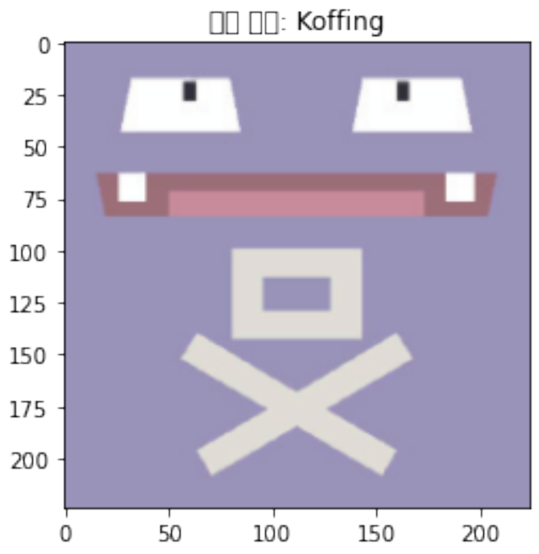
```
[예측 결과: Snorlax] (실제 정답: Snorlax)
```

```
/Users/jhs/miniforge3/lib/python3.9/site-packages/IPython/core/pylabtools
.py:151: UserWarning: Glyph 50696 (\N{HANGUL SYLLABLE YE}) missing from c
urrent font.
  fig.canvas.print_figure(bytes_io, **kw)
/Users/jhs/miniforge3/lib/python3.9/site-packages/IPython/core/pylabtools
.py:151: UserWarning: Glyph 52769 (\N{HANGUL SYLLABLE CEUG}) missing from
current font.
  fig.canvas.print_figure(bytes_io, **kw)
/Users/jhs/miniforge3/lib/python3.9/site-packages/IPython/core/pylabtools
.py:151: UserWarning: Glyph 44208 (\N{HANGUL SYLLABLE GYEOL}) missing fro
m current font.
  fig.canvas.print_figure(bytes_io, **kw)
/Users/jhs/miniforge3/lib/python3.9/site-packages/IPython/core/pylabtools
.py:151: UserWarning: Glyph 44284 (\N{HANGUL SYLLABLE GWA}) missing from
current font.
  fig.canvas.print_figure(bytes_io, **kw)
```

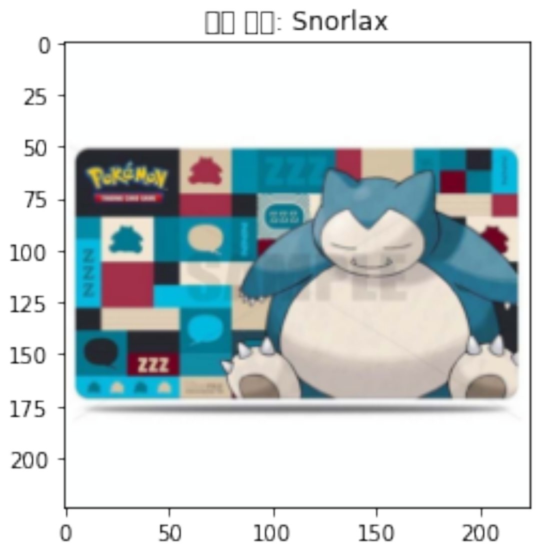
```
[예측 결과: Koffing] (실제 정답: Koffing)
```



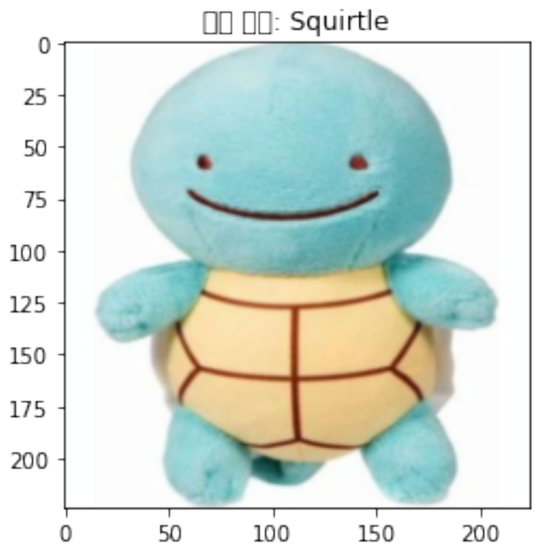
```
[예측 결과: Koffing] (실제 정답: Koffing)
```



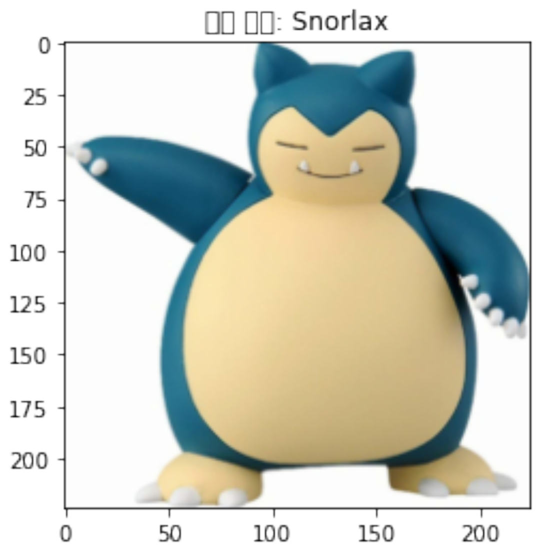
```
[예측 결과: Snorlax] (실제 정답: Snorlax)
```



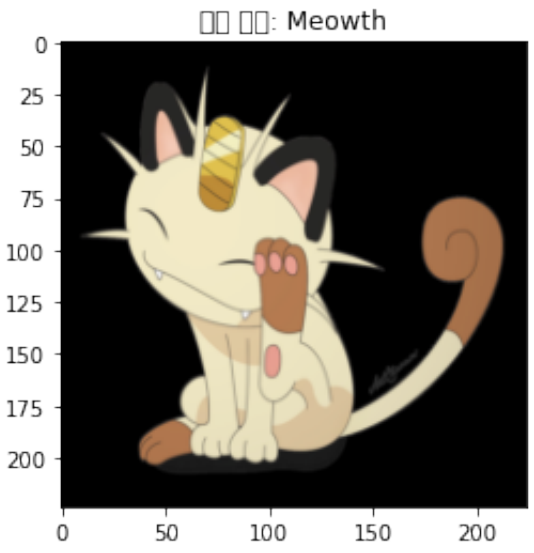
```
[예측 결과: Squirtle] (실제 정답: Squirtle)
```



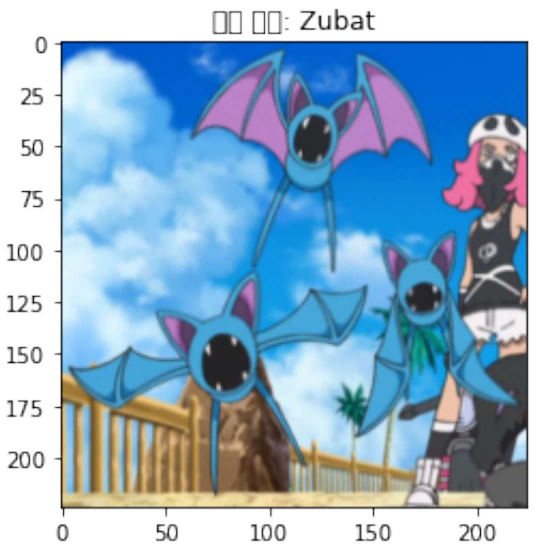
```
[예측 결과: Snorlax] (실제 정답: Snorlax)
```



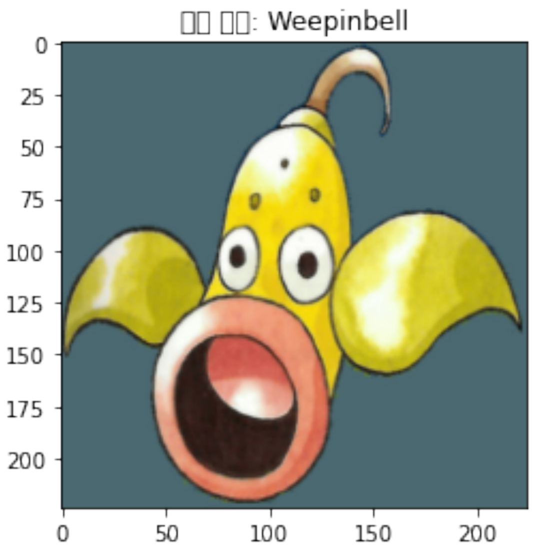
```
[예측 결과: Meowth] (실제 정답: Meowth)
```



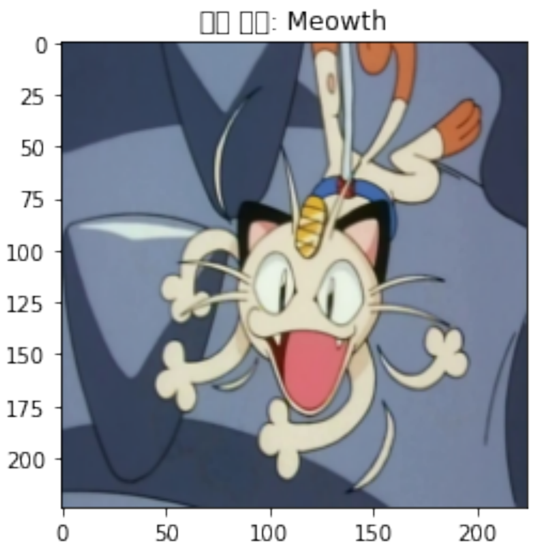
```
[예측 결과: Zubat] (실제 정답: Zubat)
```



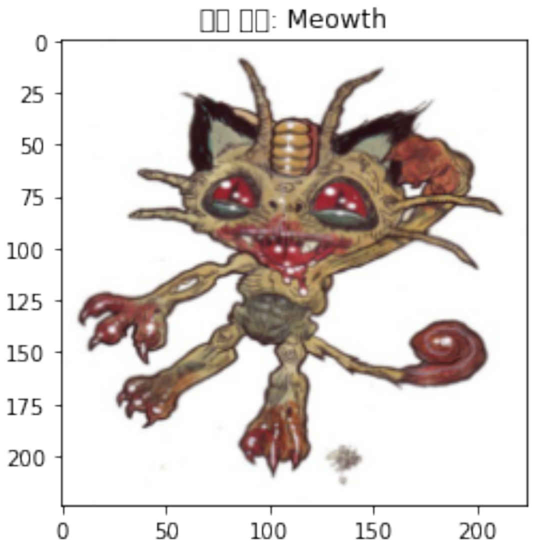
```
[예측 결과: Weepinbell] (실제 정답: Weepinbell)
```



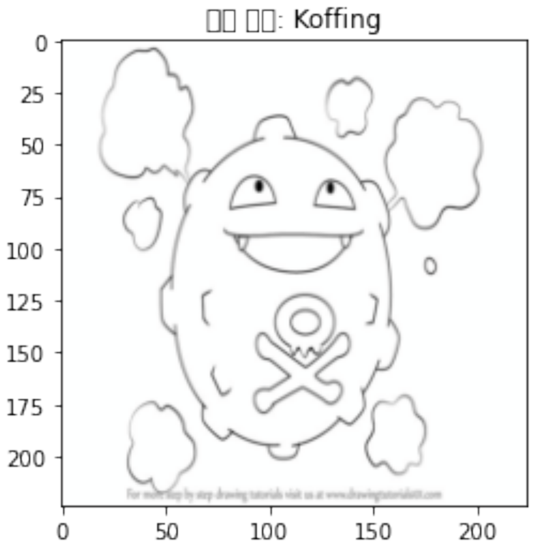
```
[예측 결과: Meowth] (실제 정답: Meowth)
```



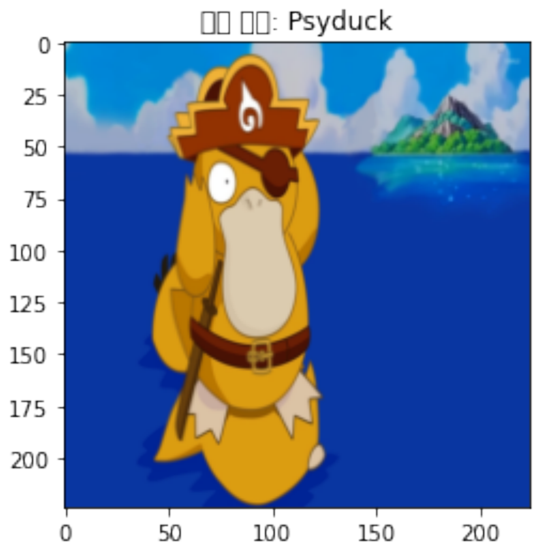
```
[예측 결과: Meowth] (실제 정답: Meowth)
```



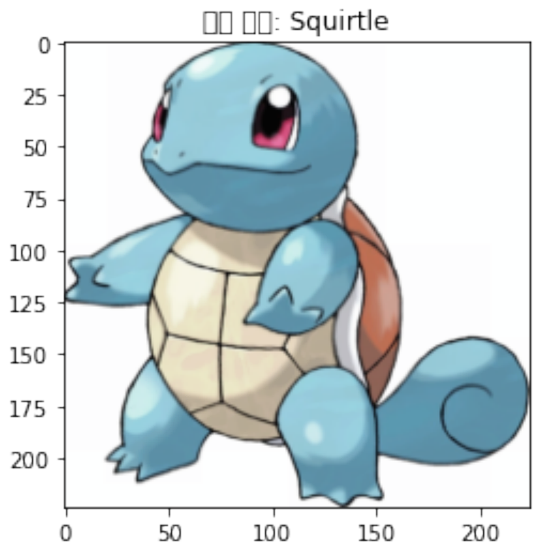
```
[예측 결과: Koffing] (실제 정답: Koffing)
```



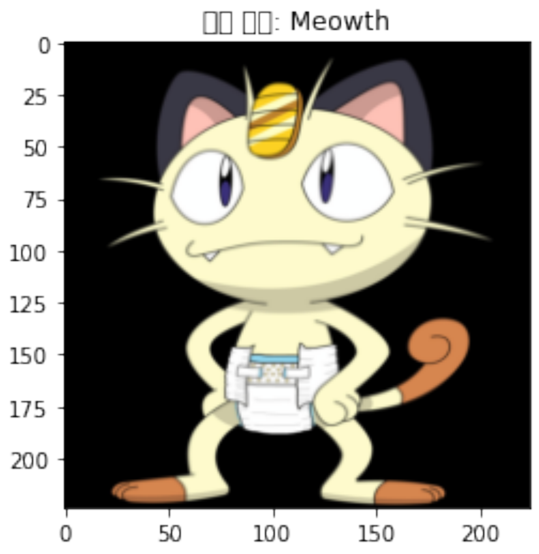
```
[예측 결과: Psyduck] (실제 정답: Psyduck)
```



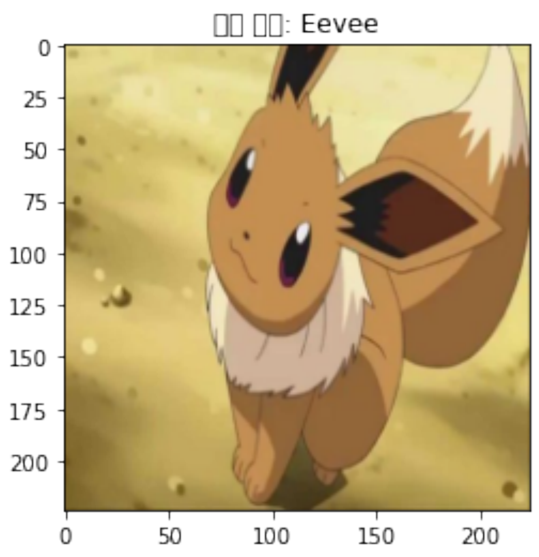
```
[예측 결과: Squirtle] (실제 정답: Squirtle)
```



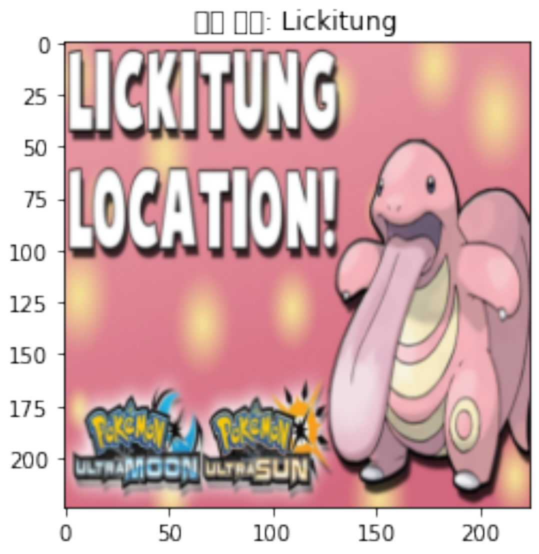
```
[예측 결과: Meowth] (실제 정답: Meowth)
```



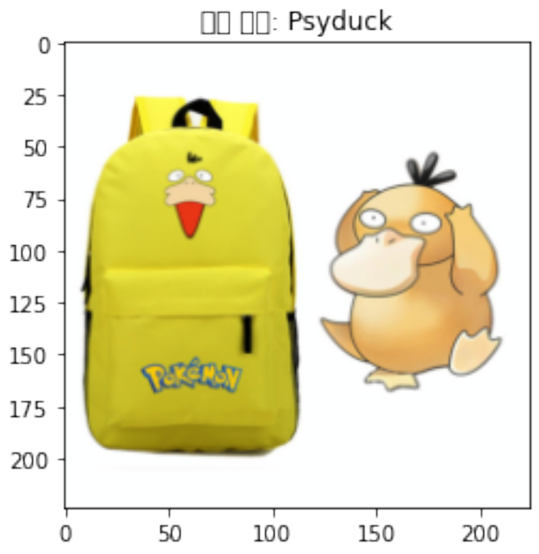
```
[예측 결과: Eevee] (실제 정답: Eevee)
```



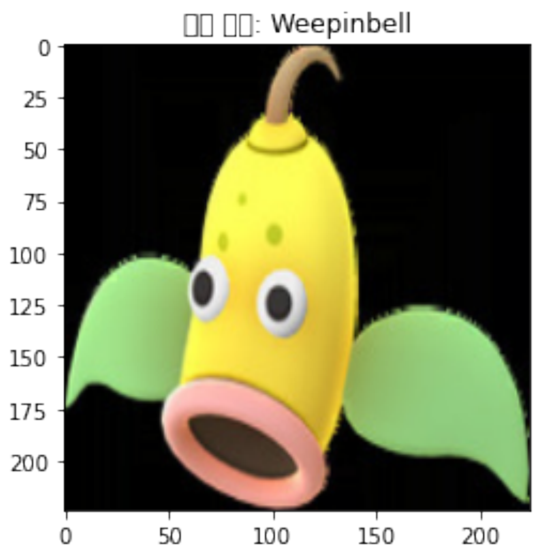
```
[예측 결과: Lickitung] (실제 정답: Lickitung)
```



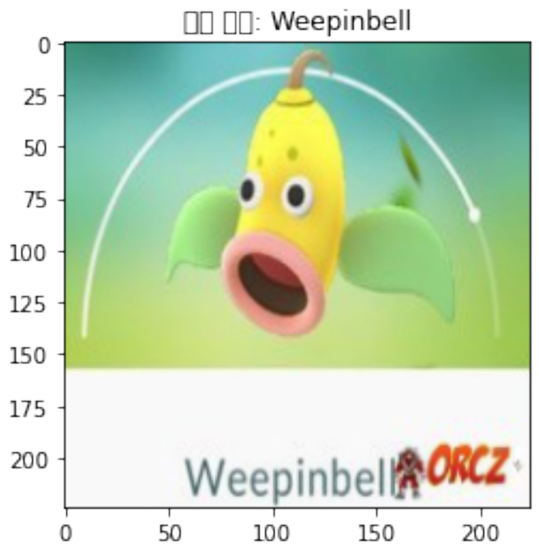
```
[예측 결과: Psyduck] (실제 정답: Psyduck)
```



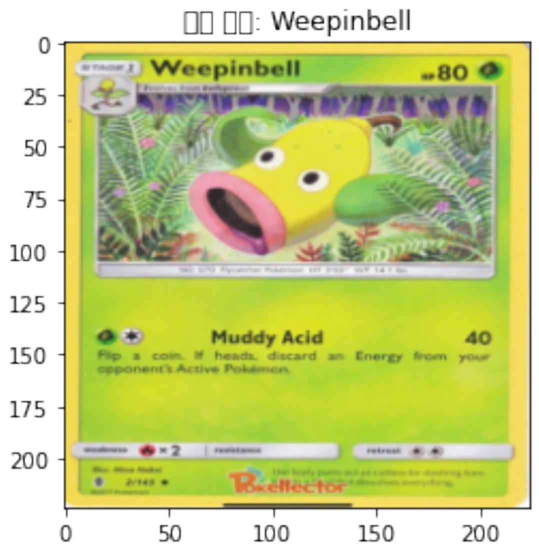
```
[예측 결과: Weepinbell] (실제 정답: Weepinbell)
```



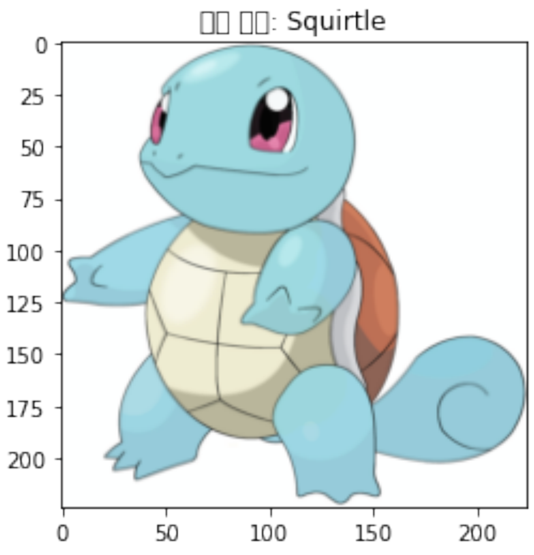
```
[예측 결과: Weepinbell] (실제 정답: Weepinbell)
```



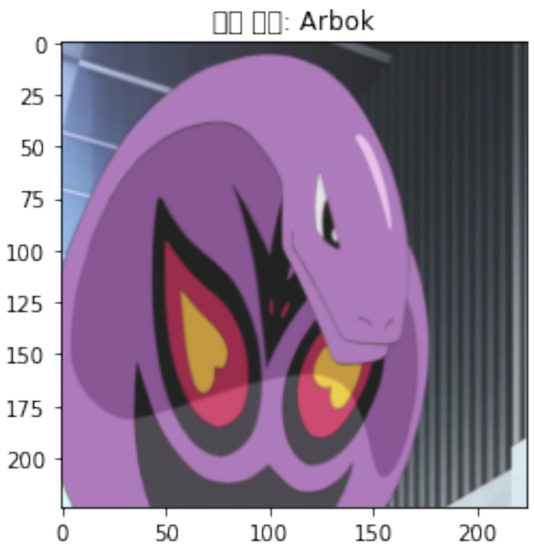
```
[예측 결과: Weepinbell] (실제 정답: Weepinbell)
```



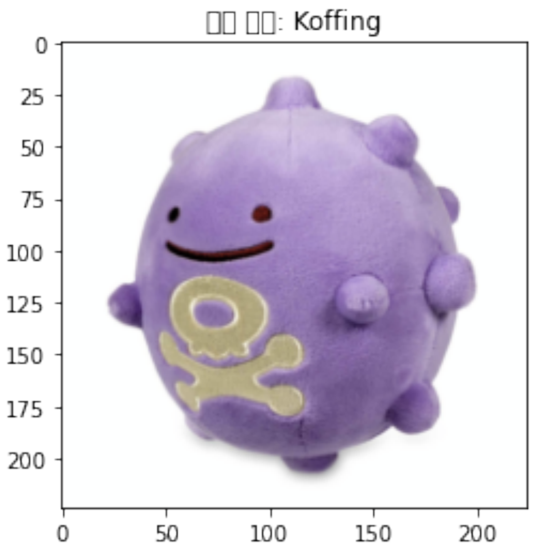
```
[예측 결과: Squirtle] (실제 정답: Squirtle)
```



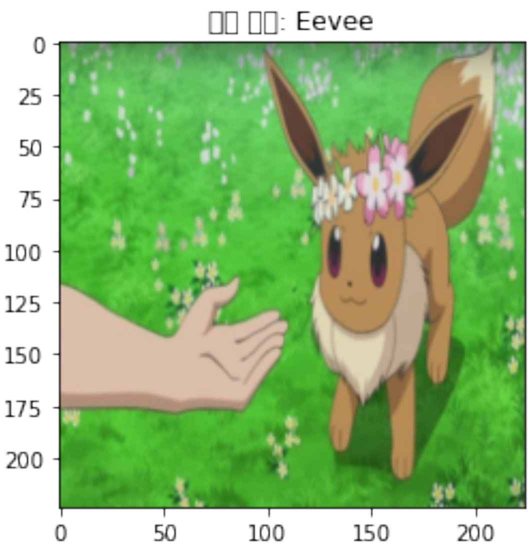
```
[예측 결과: Arbok] (실제 정답: Arbok)
```



```
[예측 결과: Koffing] (실제 정답: Koffing)
```



```
[예측 결과: Eevee] (실제 정답: Eevee)
```



```
[Test Phase] Loss: 0.2383 Acc: 94.0000% Time: 24.6567s
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: