# tf.keras.callbacks.LambdaCallback



## Class `LambdaCallback`

Inherits From: `Callback`

Defined in `tensorflow/python/keras/_impl/keras/callbacks.py`.

Callback for creating simple, custom callbacks on-the-fly.

This callback is constructed with anonymous functions that will be called at the appropriate time. Note that the callbacks expects positional arguments, as:

- `on_epoch_begin` and `on_epoch_end` expect two positional arguments: `epoch`, `logs`
- `on_batch_begin` and `on_batch_end` expect two positional arguments: `batch`, `logs`
- `on_train_begin` and `on_train_end` expect one positional argument: `logs`

### Arguments:

- `on_epoch_begin`: called at the beginning of every epoch.
- `on_epoch_end`: called at the end of every epoch.
- `on_batch_begin`: called at the beginning of every batch.
- `on_batch_end`: called at the end of every batch.
- `on_train_begin`: called at the beginning of model training.
- `on_train_end`: called at the end of model training.

Example:

```python
# Print the batch number at the beginning of every batch.
batch_print_callback = LambdaCallback(
    on_batch_begin=lambda batch,logs: print(batch))

# Stream the epoch loss to a file in JSON format. The file content
# is not well-formed JSON but rather has a JSON object per line.
import json
json_log = open('loss_log.json', mode='wt', buffering=1)
json_logging_callback = LambdaCallback(
    on_epoch_end=lambda epoch, logs: json_log.write(
        json.dumps({'epoch': epoch, 'loss': logs['loss']}) + '\n'),
    on_train_end=lambda logs: json_log.close()
)

# Terminate some processes after having finished model training.
processes = ...
cleanup_callback = LambdaCallback(
    on_train_end=lambda logs: [
        p.terminate() for p in processes if p.is_alive()])

model.fit(...,
          callbacks=[batch_print_callback,
                     json_logging_callback,
                     cleanup_callback])
```

## Methods

### __init__

```
__init__(
    on_epoch_begin=None,
    on_epoch_end=None,
    on_batch_begin=None,
    on_batch_end=None,
    on_train_begin=None,
    on_train_end=None,
    **kwargs
)
```

### on_batch_begin

```
on_batch_begin(
    batch,
    logs=None
)
```

### on_batch_end

```
on_batch_end(
    batch,
    logs=None
)
```

### on_epoch_begin

```
on_epoch_begin(
    epoch,
    logs=None
)
```

### on_epoch_end

```
on_epoch_end(
    epoch,
    logs=None
)
```

### on_train_begin

```
on_train_begin(logs=None)
```

### on_train_end

```
on_train_end(logs=None)
```

### set_model

```
set_model(model)
```

### set_params

```
set_params(params)
```