tf.contrib.bayesflow.monte_carlo.expectation_importance_sampler

```
expectation_importance_sampler(
    f,
    log_p,
    sampling_dist_q,
    z=None,
    n=None,
    seed=None,
    name='expectation_importance_sampler'
)
```

Defined in `tensorflow/contrib/bayesflow/python/ops/monte_carlo_impl.py`.

See the guide: BayesFlow Monte Carlo (contrib) > Ops

Monte Carlo estimate of `E_p[f(Z)] = E_q[f(Z) p(Z) / q(Z)]`.

With `p(z) := exp{log_p(z)}`, this `Op` returns

```
n^{-1} sum_{i=1}^n [ f(z_i) p(z_i) / q(z_i) ],  z_i ~ q,
\approx E_q[ f(Z) p(Z) / q(Z) ]
=       E_p[f(Z)]
```

This integral is done in log-space with max-subtraction to better handle the often extreme values that `f(z) p(z) / q(z)` can take on.

If `f >= 0`, it is up to 2x more efficient to exponentiate the result of `expectation_importance_sampler_logspace` applied to `Log[f]`.

User supplies either `Tensor` of samples `z`, or number of samples to draw `n`

Args:

- `f`: Callable mapping samples from `sampling_dist_q` to `Tensors` with shape broadcastable to `q.batch_shape`. For example, `f` works "just like" `q.log_prob`.
- `log_p`: Callable mapping samples from `sampling_dist_q` to `Tensors` with shape broadcastable to `q.batch_shape`. For example, `log_p` works "just like" `sampling_dist_q.log_prob`.
- `sampling_dist_q`: The sampling distribution. `tf.contrib.distributions.Distribution`. `float64` `dtype` recommended. `log_p` and `q` should be supported on the same set.
- `z`: `Tensor` of samples from `q`, produced by `q.sample` for some `n`.
- `n`: Integer `Tensor`. Number of samples to generate if `z` is not provided.
- `seed`: Python integer to seed the random number generator.
- `name`: A name to give this `Op`.

Returns:

The importance sampling estimate. `Tensor` with `shape` equal to batch shape of `q`, and `dtype` = `q.dtype`.

Last updated November 2, 2017.