tf.cond

```
cond(
    pred,
    true_fn=None,
    false_fn=None,
    strict=False,
    name=None,
    fn1=None,
    fn2=None
)
```

Defined in `tensorflow/python/ops/control_flow_ops.py`.

See the guide: Control Flow > Control Flow Operations

Return `true_fn()` if the predicate `pred` is true else `false_fn()`. (deprecated arguments)

SOME ARGUMENTS ARE DEPRECATED. They will be removed in a future version. Instructions for updating: fn1/fn2 are deprecated in favor of the true_fn/false_fn arguments.

`true_fn` and `false_fn` both return lists of output tensors. `true_fn` and `false_fn` must have the same non-zero number and type of outputs.

Note that the conditional execution applies only to the operations defined in `true_fn` and `false_fn`. Consider the following simple program:

```
z = tf.multiply(a, b)
result = tf.cond(x < y, lambda: tf.add(x, z), lambda: tf.square(y))
```

If `x < y`, the `tf.add` operation will be executed and `tf.square` operation will not be executed. Since `z` is needed for at least one branch of the `cond`, the `tf.multiply` operation is always executed, unconditionally. Although this behavior is consistent with the dataflow model of TensorFlow, it has occasionally surprised some users who expected a lazier semantics.

Note that `cond` calls `true_fn` and `false_fn` *exactly once* (inside the call to `cond`, and not at all during `Session.run()`). `cond` stitches together the graph fragments created during the `true_fn` and `false_fn` calls with some additional graph nodes to ensure that the right branch gets executed depending on the value of `pred`.

`tf.cond` supports nested structures as implemented in `tensorflow.python.util.nest`. Both `true_fn` and `false_fn` must return the same (possibly nested) value structure of lists, tuples, and/or named tuples. Singleton lists and tuples form the only exceptions to this: when returned by `true_fn` and/or `false_fn`, they are implicitly unpacked to single values. This behavior is disabled by passing `strict=True`.

Args:

- `pred`: A scalar determining whether to return the result of `true_fn` or `false_fn`.
- `true_fn`: The callable to be performed if pred is true.
- `false_fn`: The callable to be performed if pred is false.
- `strict`: A boolean that enables/disables 'strict' mode; see above.
- `name`: Optional name prefix for the returned tensors.

Returns:

Tensors returned by the call to either `true_fn` or `false_fn`. If the callables return a singleton list, the element is extracted from the list.

Raises:

- `TypeError`: if `true_fn` or `false_fn` is not callable.
- `ValueError`: if `true_fn` and `false_fn` do not return the same number of tensors, or return tensors of different types.

Example:

```
x = tf.constant(2)
y = tf.constant(5)
def f1(): return tf.multiply(x, 17)
def f2(): return tf.add(y, 23)
r = tf.cond(tf.less(x, y), f1, f2)
# r is set to f1().
# Operations in f2 (e.g., tf.add) are not executed.
```


Last updated November 2, 2017.