In [2]:
```
1 import tensorflow as tf
```

```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩h5py₩__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype fr
om `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

1. auto naming and run by name

In [7]:
```
1  # 변수 생성 및 초기화
2  v1 = tf.Variable(5.)
3  v2 = tf.Variable(15.)
4  print(v1, v2)
5
6  with tf.Session() as sess:
7      init = tf.global_variables_initializer()
8      sess.run(init)
9      print("v1 name:", v1.name)
10     print("v2 name:", v2.name)
11     v1_1, v2_1 = sess.run(["Variable_8:0", "Variable_9:0"])
12     print("v1 value: %f" % (v1_1))
13     print("v2 value: %f" % (v2_1))
14
```

```
<tf.Variable 'Variable_10:0' shape=() dtype=float32_ref> <tf.Variable 'Variable_11:0' shape=() dtype=float32_ref>
v1 name: Variable_10:0
v2 name: Variable_11:0
v1 value: 5.000000
v2 value: 15.000000
```

02. get_Variable

- get_variable()는 원칙적으로 해당 name field값이 있는 tensor가 있는지를 먼저 확인 후, 새로운 tensor를 생성한다.

In [11]:
```
v3 = tf.Variable(3.,name="v3")  # v3으로 이름 붙이기
v4 = tf.Variable(4.,name="v4")
print(v3)
print(v4)
```

```
<tf.Variable 'v3_2:0' shape=() dtype=float32_ref>
<tf.Variable 'v4_2:0' shape=() dtype=float32_ref>
```

In [14]:
```
v5 = tf.get_variable("v7",1,initializer=tf.constant_initializer(5.))
v6 = tf.get_variable("v8",1,initializer=tf.constant_initializer(6.))
print(v5)
print(v6)  # 존재하는 name tensor가 있다면 에러를 띄운다.
```

```
<tf.Variable 'v7:0' shape=(1,) dtype=float32_ref>
<tf.Variable 'v8:0' shape=(1,) dtype=float32_ref>
```

In [15]:
```
with tf.Session() as sess:
    init = tf.global_variables_initializer()
    sess.run(init)

    print("v3 name:", v3.name)
    print("v4 name:", v4.name)
    print("v5 name:", v5.name)
    print("v6 name:", v6.name)
    v3_1, v4_1, v5_1, v6_1 = sess.run(["v3:0", "v4:0", "v5:0", "v6:0"])

    print("v3_1 value: %f" % (v3_1)  )
    print("v4_1 value: %f" % (v4_1)  )
    print("v5_1 value: %f" % (v5_1)  )
    print("v6_1 value: %f" % (v6_1)  )
```

```
v3 name: v3_2:0
v4 name: v4_2:0
v5 name: v7:0
v6 name: v8:0
v3_1 value: 3.000000
v4_1 value: 4.000000
v5_1 value: 5.000000
v6_1 value: 6.000000
```

03. name_scope

- get_variable는 name이 있는지 없는지 확인

In [3]:
```
with tf.name_scope("scope1"):
    v3 = tf.Variable(3., name="v3")
    v4 = tf.Variable(4., name="v4")
    v3_1 = tf.get_variable("v3",1,initializer=tf.constant_initializer(3.))
    v4_1 = tf.get_variable("v4",1,initializer=tf.constant_initializer(4.))

```

```
v3 name Variable: scope1/v3:0
v4 name Variable: scope1/v4:0
v3 name get_variable: v3:0
v4 name get_variable: v4:0
```

In [6]:
```
print ("v3 name Variable:", v3.name)
print ("v4 name Variable:", v4.name)
print ("v3 name get_variable:", v3_1.name)
print ("v4 name get_variable:", v4_1.name)
```

```
v3 name Variable: scope1/v3:0
v4 name Variable: scope1/v4:0
v3 name get_variable: v3:0
v4 name get_variable: v4:0
```

04. variable_scope

- tf.get_varialbe()에게도 tf.Variable() 처럼 name_scope()영향을 동일하게 주기 위한 것이 tf.variable_scope()이다.

In [7]:
```
with tf.variable_scope("scope2"):
    v1 = tf.get_variable("v1",1,initializer=tf.constant_initializer(1.))
    v2 = tf.get_variable("v2",1,initializer=tf.constant_initializer(2.))
    v3 = tf.Variable(3.,name="v3")
    v4 = tf.Variable(3.,name="v4")

    print("v1 name:", v1.name)
    print("v2 name:", v2.name)
    print("v3 name:", v3.name)
    print("v4 name:", v4.name)

```

```
v1 name: scope2/v1:0
v2 name: scope2/v2:0
v3 name: scope2/v3:0
v4 name: scope2/v4:0
```

동일한 name를 하게 되면 에러 발생

In [8]:
```
1 with tf.variable_scope("scope2"):
2     v1 = tf.get_variable("v1",1,initializer=tf.constant_initializer(1.))
3     v2 = tf.get_variable("v2",1,initializer=tf.constant_initializer(2.))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-487d1198baf7> in <module>()
      1 with tf.variable_scope("scope2"):
----> 2     v1 = tf.get_variable("v1",1,initializer=tf.constant_initializer(1.))
      3     v2 = tf.get_variable("v2",1,initializer=tf.constant_initializer(2.))

~₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩ops₩variable_scope.py in get_variable(name, shape, dtype, initializer, regular
izer, trainable, collections, caching_device, partitioner, validate_shape, use_resource, custom_getter, constraint)
   1315       partitioner=partitioner, validate_shape=validate_shape,
   1316       use_resource=use_resource, custom_getter=custom_getter,
-> 1317       constraint=constraint)
   1318 get_variable_or_local_docstring = (
   1319     """%s

~₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩ops₩variable_scope.py in get_variable(self, var_store, name, shape, dtype, ini
tializer, regularizer, reuse, trainable, collections, caching_device, partitioner, validate_shape, use_resource, custom_gette
r, constraint)
   1077           partitioner=partitioner, validate_shape=validate_shape,
   1078           use_resource=use_resource, custom_getter=custom_getter,
-> 1079           constraint=constraint)
   1080
   1081   def _get_partitioned_variable(self,

~₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩ops₩variable_scope.py in get_variable(self, name, shape, dtype, initializer, r
egularizer, reuse, trainable, collections, caching_device, partitioner, validate_shape, use_resource, custom_getter, constrain
t)
    423           caching_device=caching_device, partitioner=partitioner,
    424           validate_shape=validate_shape, use_resource=use_resource,
--> 425           constraint=constraint)
    426
    427   def _get_partitioned_variable(

~₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩ops₩variable_scope.py in _true_getter(name, shape, dtype, initializer, regular
izer, reuse, trainable, collections, caching_device, partitioner, validate_shape, use_resource, constraint)
    392           trainable=trainable, collections=collections,
    393           caching_device=caching_device, validate_shape=validate_shape,
```

```
--> 394           use_resource=use_resource, constraint=constraint)
    395
    396     if custom_getter is not None:

~₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩ops₩variable_scope.py in _get_single_variable(self, name, shape, dtype, initial
izer, regularizer, partition_info, reuse, trainable, collections, caching_device, validate_shape, use_resource, constraint)
    731                          "reuse=tf.AUTO_REUSE in VarScope? "
    732                          "Originally defined at:₩n₩n%s" % (
--> 733                              name, "".join(traceback.format_list(tb))))
    734       found_var = self._vars[name]
    735       if not shape.is_compatible_with(found_var.get_shape()):

ValueError: Variable scope2/v1 already exists, disallowed. Did you mean to set reuse=True or reuse=tf.AUTO_REUSE in VarScope? Orig
inally defined at:

  File "C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩framework₩ops.py", line 1718, in __init__
    self._traceback = self._graph._extract_stack()  # pylint: disable=protected-access
  File "C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩framework₩ops.py", line 3392, in create_op
    op_def=op_def)
  File "C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩framework₩op_def_library.py", line 787, in _apply_op_helper
    op_def=op_def)
```

In [9]:
```
with tf.variable_scope("scope3"):
    v1 = tf.get_variable("v1",1,initializer=tf.constant_initializer(1.))
    v2 = tf.get_variable("v2",1,initializer=tf.constant_initializer(2.))
```

In [10]:
```
    print("v1 name:", v1.name)
    print("v2 name:", v2.name)
```

```
v1 name: scope3/v1:0
v2 name: scope3/v2:0
```

In [ ]:
```

```