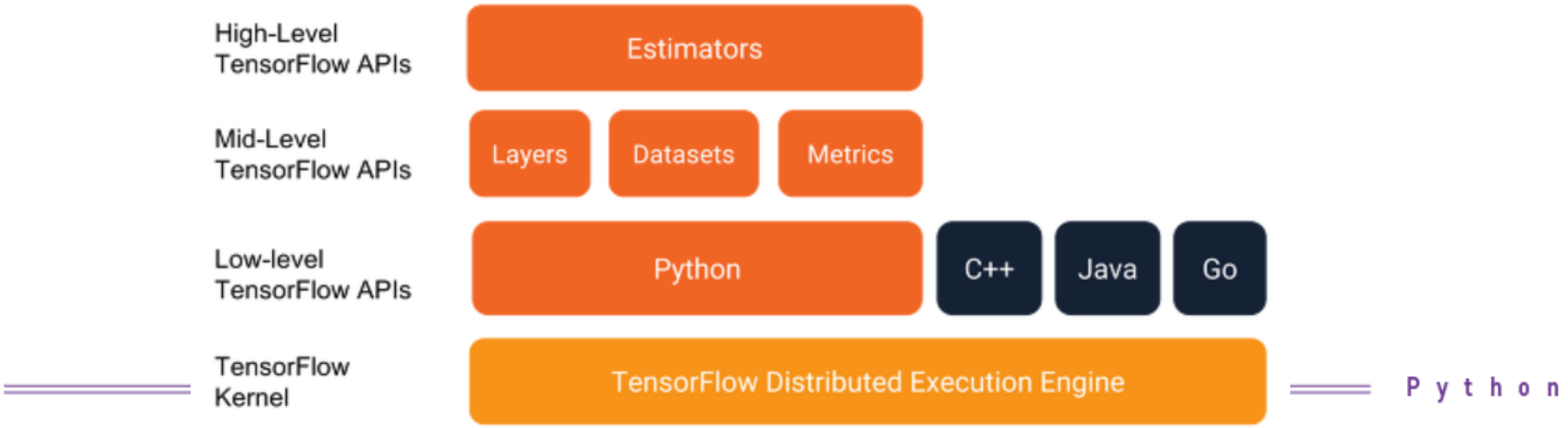
텐서플로 개요

참고 https://excelsior-cjh.tistory.com/148 [EXCELSIOR]

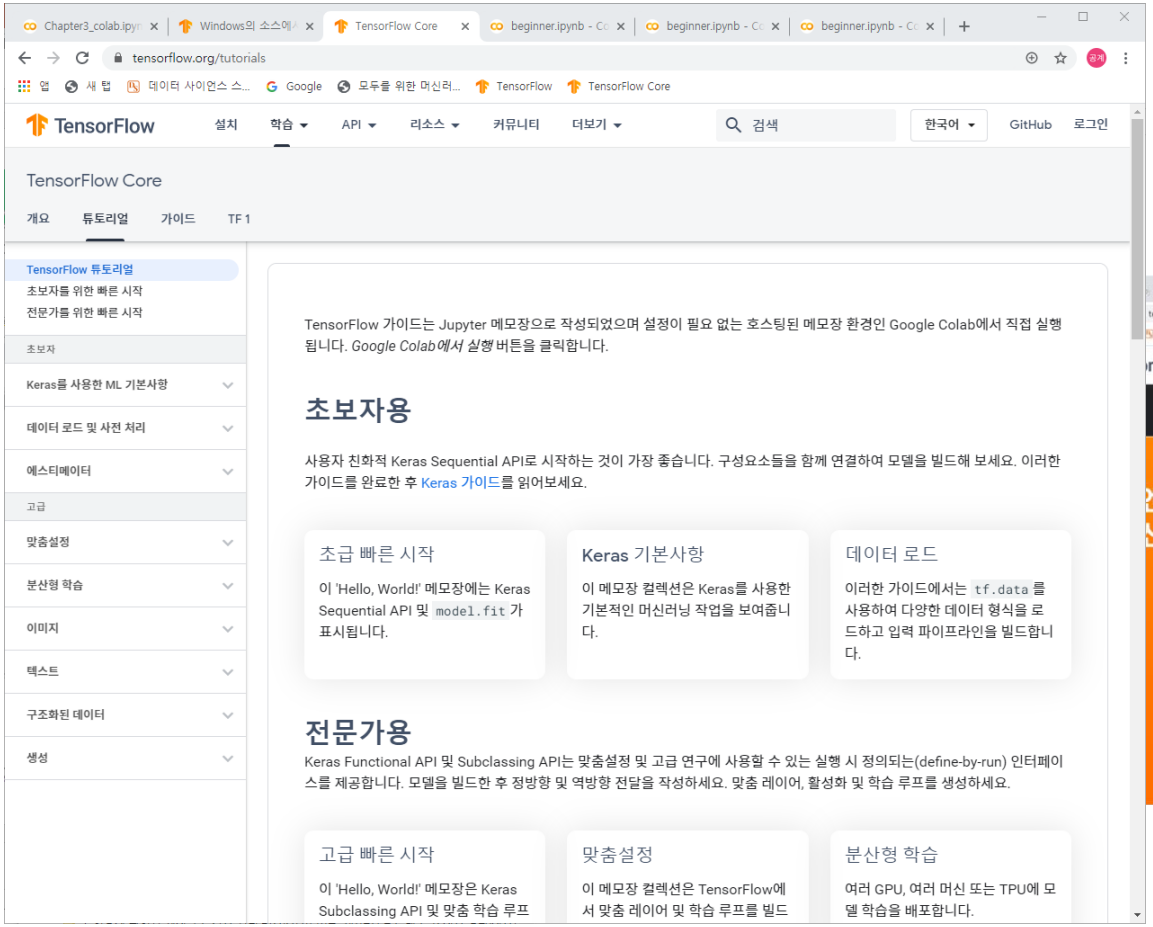
텐서플로(TensorFlow) 개요

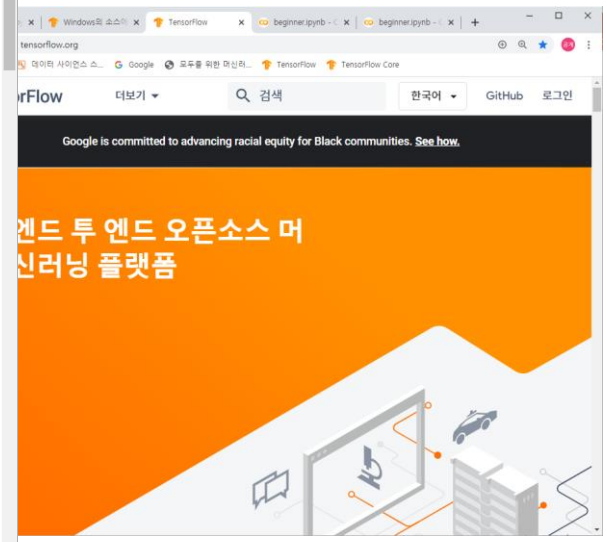
- **구글(Google)에서 만든 라이브러리**
  - www.tensorflow.org
  - 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공
  - 초보자 및 전문가에게 데스크톱, 모바일, 웹, 클라우드 개발용 API를 제공
  - 연구 및 프로덕션용 오픈소스 딥러닝 라이브러리
- **구현 및 사용**
  - 텐서플로 자체는 기본적으로 C++로 구현
  - Python, Java, Go 등 다양한 언어를 지원
  - 파이썬을 최우선으로 지원하며 대부분의 편한 기능들이 파이썬 라이브러리로만 구현되어 있어 Python에서 개발하는 것이 편함



텐서플로 홈페이지

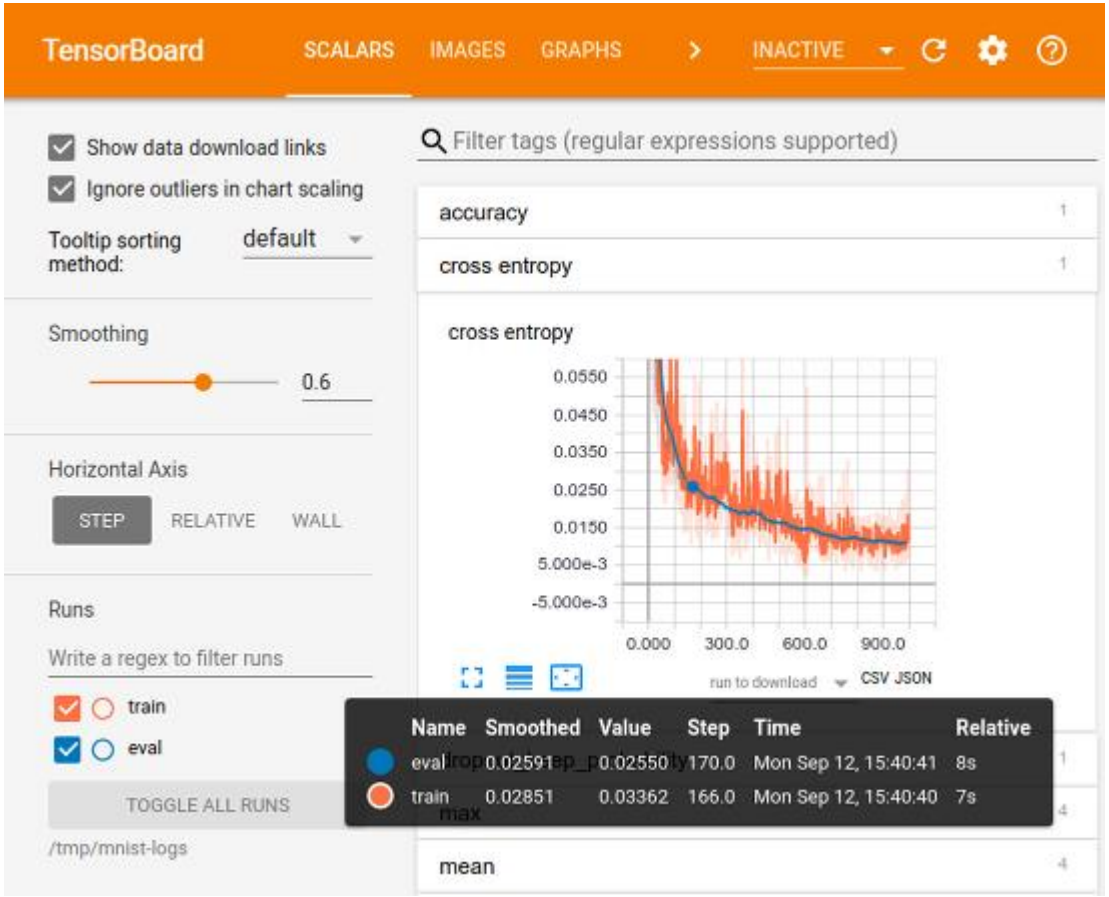
- **튜토리얼**
  - https://www.tensorflow.org/tutorials





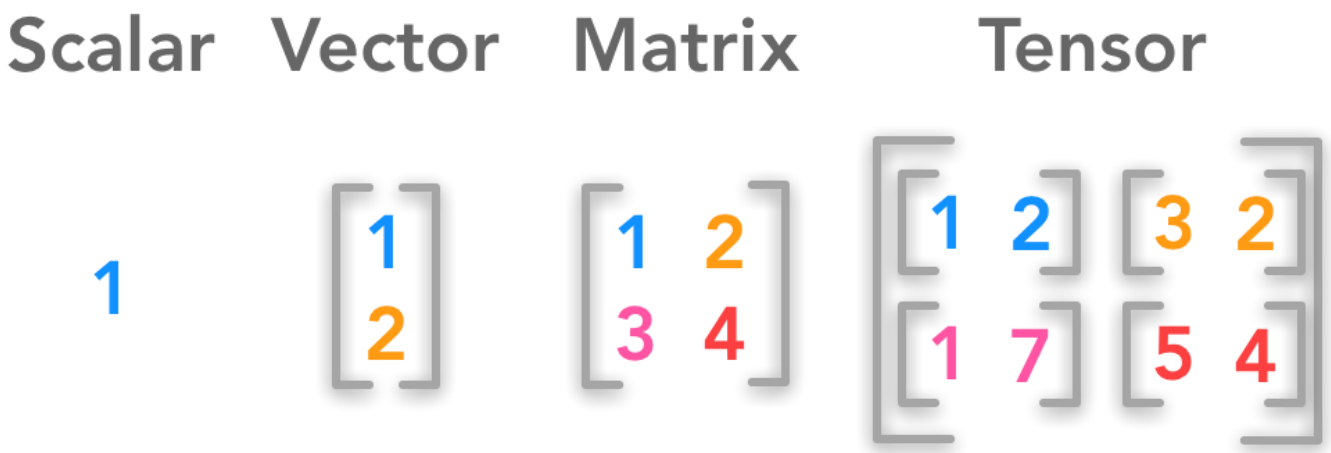
텐서보드(TensorBoard)

- **브라우저에서 실행 가능한 시각화 도우미**
  - 딥러닝 학습 과정을 추적하는데 유용하게 사용



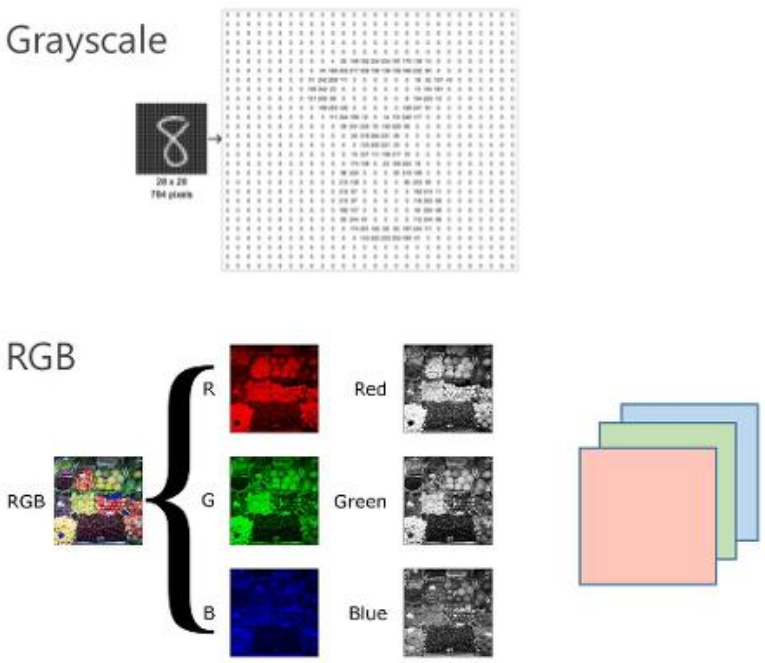
텐서 개요

- **Tensor(텐서): 모든 데이터**
  - 딥러닝에서 데이터를 표현하는 방식
    - **0-D 텐서 : 스칼라**
    - **1-D 텐서 : 벡터**
    - **2-D 텐서 : 행렬 등**
  - n차원 행렬(배열)
    - **텐서는 행렬로 표현할 수 있는 n차원 형태의 배열을 높은 차원으로 확장**



텐서의 사례

- **스칼라: 차원이 없는 텐서**
  - 10
- **벡터 값: 1차원 텐서**
  - [10, 20, 30]
- **2차원 행렬: 2차원 텐서**
  - 회색조(grayscale) 이미지
    - **하나의 채널(channel)에 2차원 행렬(배열)로 표현**
  - [[1, 2, 3], [4, 5, 6]]
- **텐서: n차원 행렬**
  - 텐서의 차원을 텐서의 rank(순위)라 함
  - RGB 이미지
    - **R(ed), G(reen), B(lue) 각 3개의 채널마다 2차원 행렬(배열)로 표현하는데, 이를 텐서(3차원의 값을 가지는 배열)로 표현**
  - [[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]]



TensorFlow 계산 과정

- **TensorFlow에서 텐서(Tensor) 계산 과정**
  - 모두 그래프(Graph)라고 부르는 객체 내에 저장되어 실행
  - 그래프를 계산하려면 외부 컴퓨터에 이 그래프 정보를 전달하고 그 결과값을 받아야 함
- **Session**
  - 이 통신과정을 담당하는 것이 세션(Session)이라고 부르는 객체
  - 생성, 사용, 종료 과정이 필요
  - 세션 생성
    - **Session 객체 생성**
  - 세션 사용
    - **run 메서드에 그래프를 입력하면 출력 값을 계산하여 반환**
  - 세션 종료
    - **close 메서드.**
    - **with 문을 사용하면 명시적으로 호출 불필요**

```
x = tf.constant(3)
y = x**2

sess = tf.Session()
print(sess.run(x))
print(sess.run(y))
sess.close()
```

데이터 흐름 그래프(dataflow graph)(1)

- **TensorFlow에서 계산**
  - 데이터 흐름 그래프(dataflow graph)로 이루어 짐
  - 텐서 형태의 데이터들이 딥러닝 모델을 구성하는 연산들의 그래프를 따라 흐르면서 연산이 일어남
- **Tensor + DataFlow**
  - 딥러닝에서 데이터를 의미하는 Tensor 와 DataFlow Graph를 따라 연산이 수행되는 형태(Flow)를 합쳐 TensorFlow란 이름이 나오게 됨

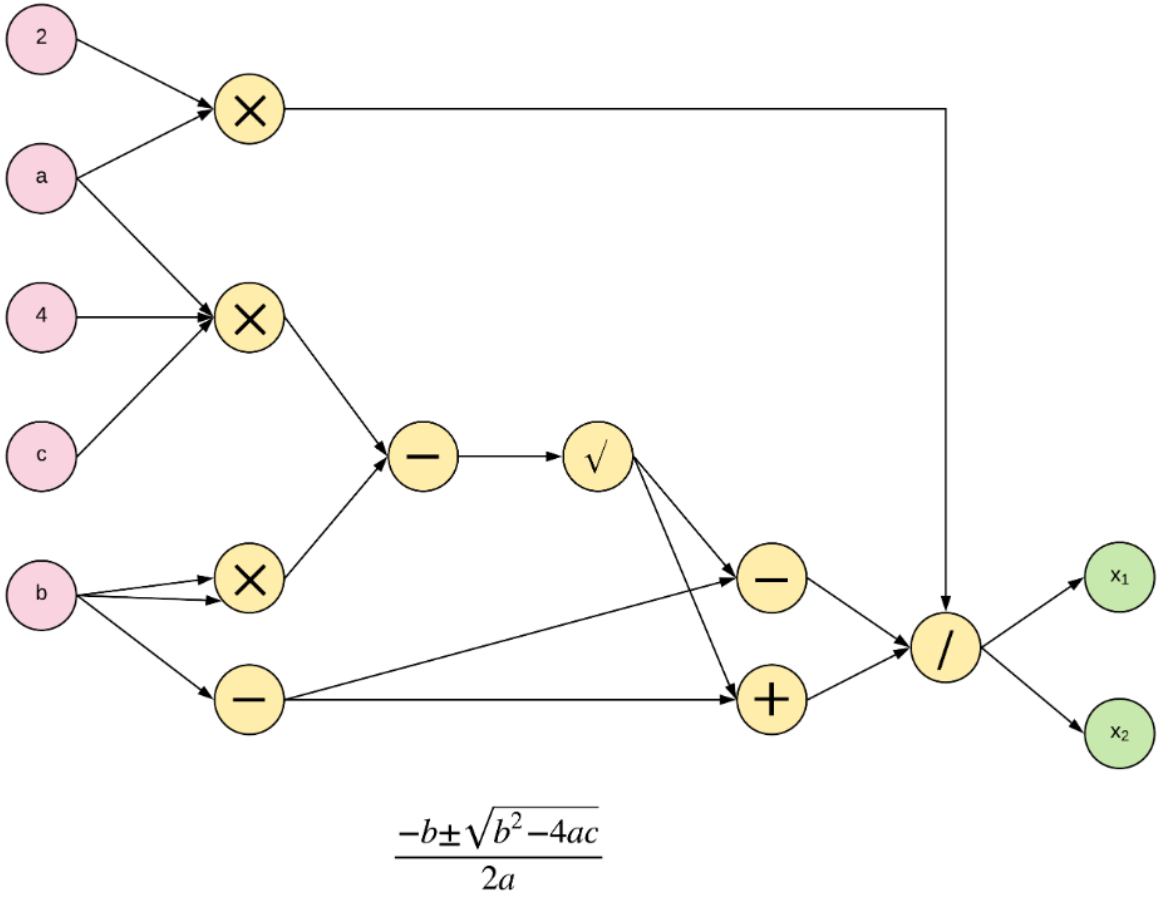
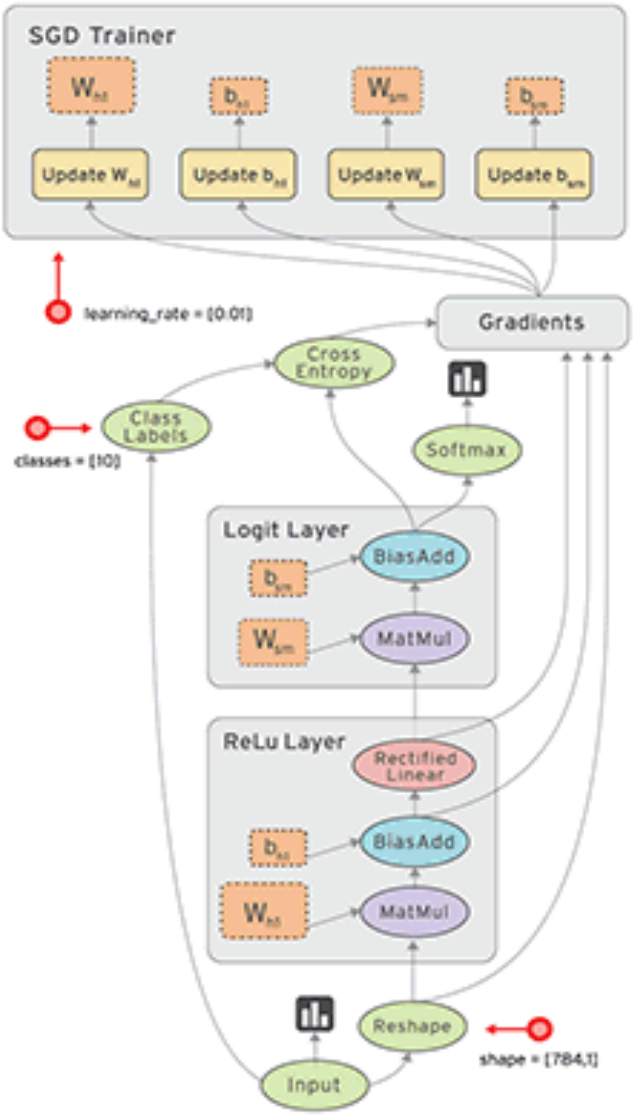


그림 1 : 2 차 공식을 사용하여 2 차 표현식의 근을 계산하기위한 계산 그래프.

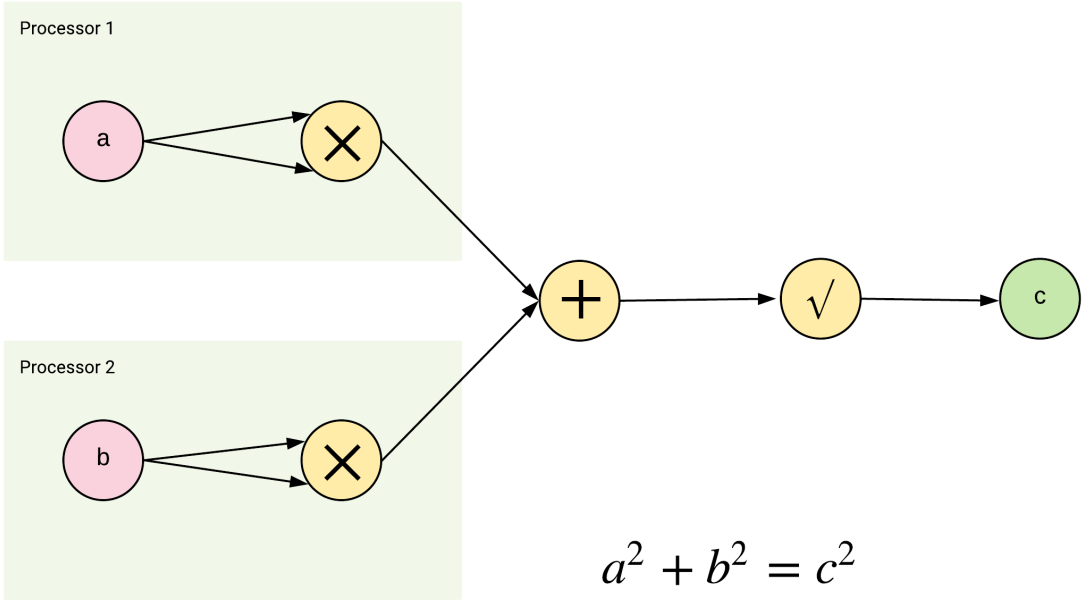
데이터 흐름 그래프(dataflow graph)(2)

- **노드(nodes)**
  - 수리적 계산
- **엣지(edges)**
  - 자료(텐서)의 이동
- **TensorFlow 프로그램**
  - 텐서로 모든 데이터를 표현
    - **모델에 사용하려는 모든 유형의 데이터를 Tensor에 저장**
      - **간단히 말해서, Tensor는 다차원 배열**
      - **0-D 텐서 : 스칼라**
      - **1-D 텐서 : 벡터**
      - **2-D 텐서 : 매트릭스 등**
  - TensorFlow는 단순히 계산 그래프에서 Tensor의 흐름을 표현



TensorFlow는 대규모 기계 학습 구축에 적합

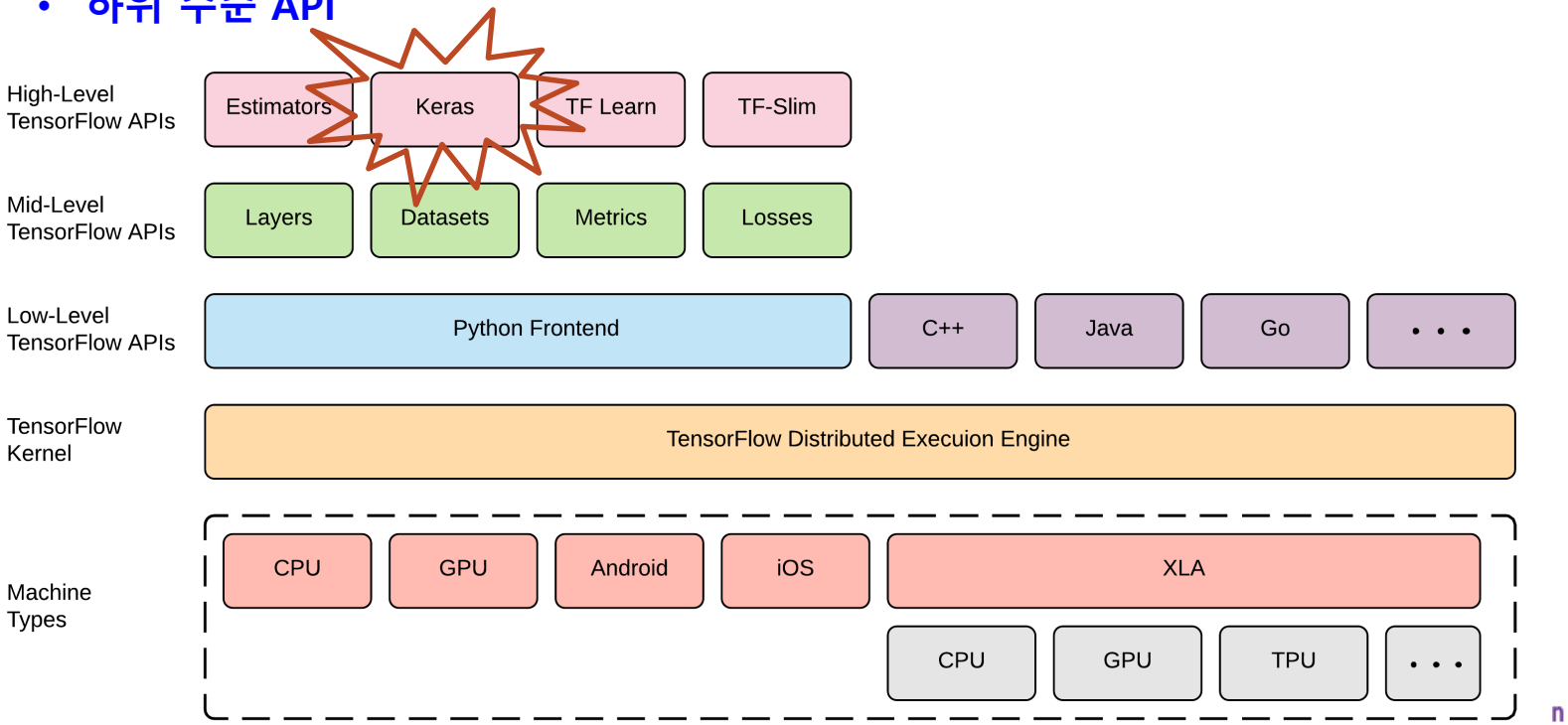
- **데이터와 연산이 이벤트 그래프로 설계되면**
  - 그래프를 개척하고 여러 프로세서에서 병렬로 그래프의 다른 부분을 실행 가능
    - **CPU, GPU 또는 TPU 사용 가능**
- **TensorFlow**
  - 대규모 기계 학습 제품을 구축하기에 적합



$$a^2 + b^2 = c^2$$

TensorFlow API 탐색

- **TensorFlow API 계층**
  - TensorFlow 딥 러닝 모델 구축 작업은 서로 다른 API 수준을 사용하여 해결
    - **고급 API**
      - **Keras나 TF-Slim 과 같은 추상화 라이브러리를 제공하여 저수준 텐서플로 라이브러리에 대해 손쉽게 고수준 접근이 가능하게 해줌**
    - **중급 API**
    - **하위 수준 API**



텐서플로 vs 케라스

- 케라스. 덴서플로 뭐가 좋아요?

Hello World! 코딩

- **설치 확인**

```
import tensorflow as tf
tf.__version__
'1.0.0'
```

- **첫 코딩**
  - 상수 노드 생성
  - Session을 만들어
  - 실행

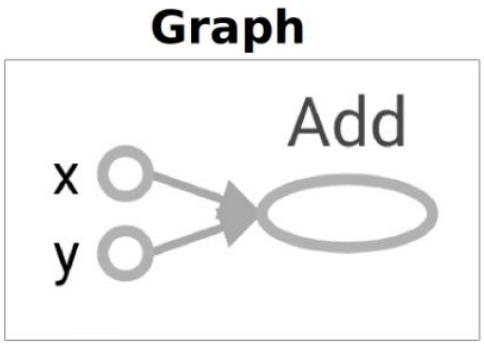
```
import tensorflow as tf

hello = tf.constant("Hello World!")

sess = tf.Session()
print(sess.run(hello))

sess.close()
```

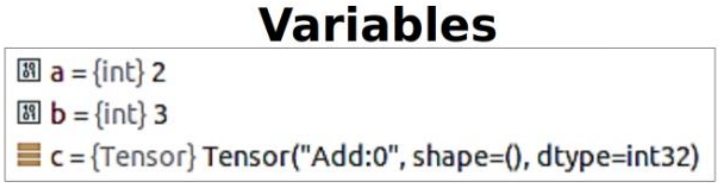
데이터플로 그래프의 이해: 그래프 생성

- **Add 노드**

In [36]:
```
import tensorflow as tf

a = 2
b = 3
c = tf.add(a, b, name='Add')

print(a, b)
print(c)
```

```
2 3
Tensor("Add_14:0", shape=(), dtype=int32)
```





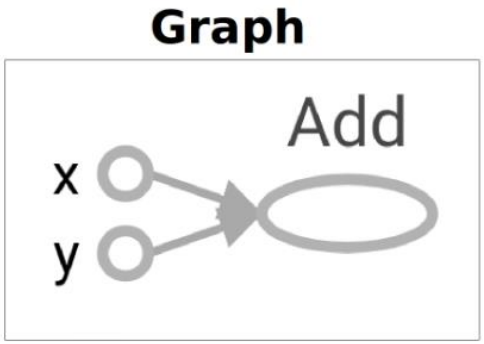
https://www.easy-tensorflow.com/tf-tutorials/basics/graph-and-session

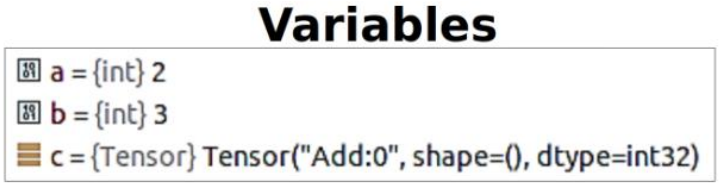
데이터플로 그래프의 이해: 그래프 실행

- **Session()을 생성해 실행**

```
In [38]: import tensorflow as tf

         a = 2
         b = 3
         c = tf.add(a, b, name='Add')

         sess = tf.Session()
         print(sess.run(c))
         sess.close()
```

```
5
```

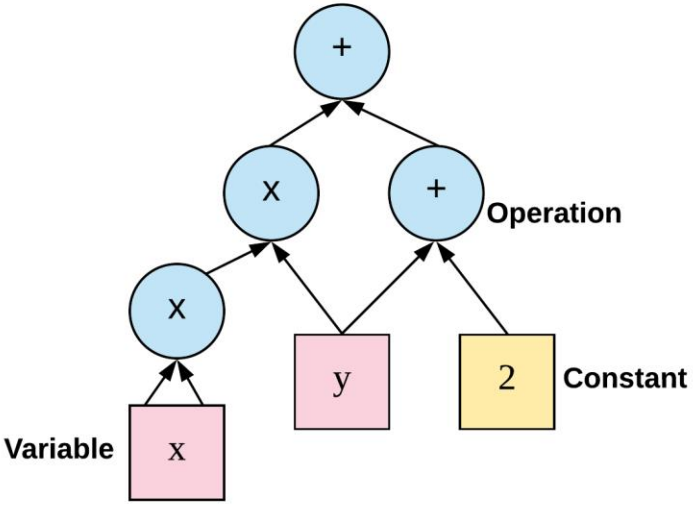




https://www.easy-tensorflow.com/tf-tutorials/basics/graph-and-session

데이터플로 그래프의 이해(1)

- **$f(x, y) = x^2 \times y + y + 2$**
  - X = 2, y = 3인 경우
    - **17**



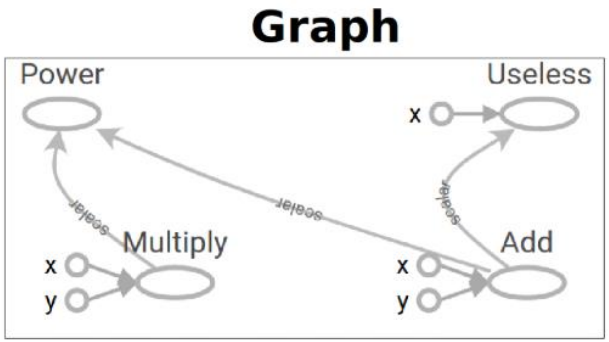
In [44]:
```
import tensorflow as tf

x = 2
y = 3
pow1 = tf.pow(x, 2)
mul1 = tf.multiply(pow1, y)
add1 = tf.add(y, 2)
add2 = tf.add(mul1, add1)

with tf.Session() as sess:
    print(sess.run(add2))
```

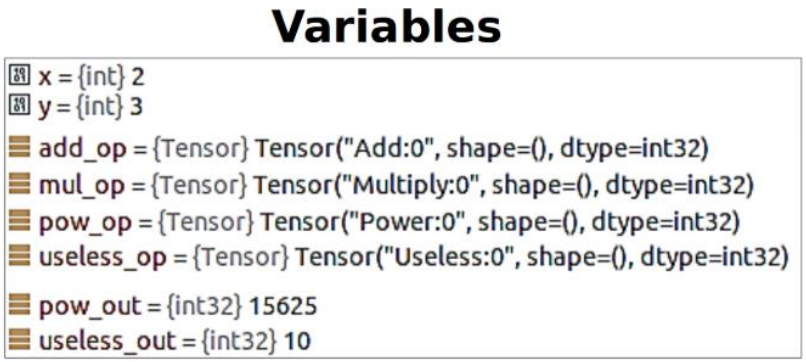
```
17
```

https://www.easy-tensorflow.com/tf-tutorials/basics/graph-and-session

데이터플로 그래프의 이해(2)

- 그래프





- 다음 코드의 결과는?

In [39]:
```
import tensorflow as tf

x = 2
y = 3
add_op = tf.add(x, y, name='Add')
mul_op = tf.multiply(x, y, name='Multiply')
pow_op = tf.pow(add_op, mul_op, name='Power')
useless_op = tf.multiply(x, add_op, name='Useless')

with tf.Session() as sess:
    print(sess.run(pow_op))
    print(sess.run(useless_op))
```

```
15625
10
```

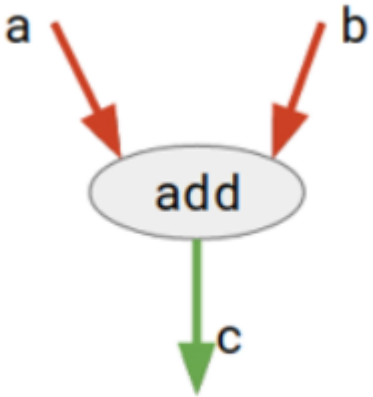
a + b

- 노드 출력

```
n1 = tf.constant(7)
n2 = tf.constant(2)

sess = tf.Session()
print(sess.run([n1, n2]))
sess.close()
```

- a + b

```
a = tf.constant(7)
b = tf.constant(2)
addnode = tf.add(a, b)

sess = tf.Session()
print(sess.run(addnode))
sess.close()
```



computation graph

```
a = tf.constant(7)
b = tf.constant(2)
addnode = tf.add(a, b)

with tf.Session() as sess:
    print(sess.run(addnode))
```

TensorFlow 기계

- **계산 정의(그래프 생성)와 실행 정의(세션의 실행)를 분리**
  - 계산 정의는 Python의 함수 정의라 할 수 있음
    - **즉 실행하지 않으면 결과는 없음**

2 feed data and run graph (operation)
sess.run (op)

1 Build graph using TensorFlow operations

3 update variables in the graph (and return values)

WWW.MATHWAREHOUSE.COM

Tensor Ranks, Shapes, and Types(1)

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Rank	Math entity	Python example
0	Scalar (magnitude only)	`s = 483`
1	Vector (magnitude and direction)	`v = [1.1, 2.2, 3.3]`
2	Matrix (table of numbers)	`m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`
3	3-Tensor (cube of numbers)	`t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]`
n	n-Tensor (you get the idea)	`....`

https://www.tensorflow.org/programmers_guide/dims_types

Tensor Ranks, Shapes, and Types(1)

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.

https://www.quora.com/When-should-I-use-tf-float32-vs-tf-float64-in-TensorFlow

텐서플로 난수

- **tf.random_normal**
  - 정규분포의 난수를 주어진 형태로 발생

In [36]:
```
r1 = tf.random_normal([1])
r2 = tf.random_normal([1])
print(r1, r2)

with tf.Session() as sess:
    n1, n2 = sess.run([r1, r2])
    print(n1, n2)
```

```
Tensor("random_normal_17:0", shape=(1,), dtype=float32) Tensor("random_normal_18:0",
shape=(1,), dtype=float32)
[0.6607665] [-1.6369933]
```

In [37]:
```
r1 = tf.random_normal([1, 3])
r2 = tf.random_normal([1, 5])
print(r1, r2)

with tf.Session() as sess:
    n1, n2 = sess.run([r1, r2])
    print(n1, n2)
```

```
Tensor("random_normal_19:0", shape=(1, 3), dtype=float32) Tensor("random_normal_20:
0", shape=(1, 5), dtype=float32)
[[-0.55198944  0.90757304 -0.77650166]] [[-2.4719653   0.17210001 -0.667049   -0.6740
342   0.16063188]]
```

텐서플로 API

- **API**
  - https://www.tensorflow.org/api_docs/python/tf

Modules

`audio` module: Public API for tf.audio namespace.

`autodiff` module: Public API for tf.autodiff namespace.

`autograph` module: Conversion of plain Python into TensorFlow graph code.

`bitwise` module: Operations for manipulating the binary representations of integers.

`compat` module: Compatibility functions.

`config` module: Public API for tf.config namespace.

`data` module: `tf.data.Dataset` API for input pipelines.

`debugging` module: Public API for tf.debugging namespace.

`distribute` module: Library for running a computation across multiple devices.

`dtypes` module: Public API for tf.dtypes namespace.

`errors` module: Exception types for TensorFlow errors.

`estimator` module: Estimator: High level tools for working with models.

`experimental` module: Public API for tf.experimental namespace.

`feature_column` module: Public API for tf.feature_column namespace.

텐서플로 class와 function

Classes

`class AggregationMethod` : A class listing aggregation methods used to combine gradients.

`class CriticalSection` : Critical section.

`class DType` : Represents the type of the elements in a `Tensor`.

`class DeviceSpec` : Represents a (possibly partial) specification for a TensorFlow device.

`class GradientTape` : Record operations for automatic differentiation.

`class Graph` : A TensorFlow computation, represented as a dataflow graph.

`class IndexedSlices` : A sparse representation of a set of tensor slices at given indices.

`class IndexedSlicesSpec` : Type specification for a `tf.IndexedSlices`.

`class Module` : Base neural network module class.

`class Operation` : Represents a graph node that performs computation on tensors.

`class OptionalSpec` : Represents an optional potentially containing a structured value.

`class RaggedTensor` : Represents a ragged tensor.

`class RaggedTensorSpec` : Type specification for a `tf.RaggedTensor`.

Functions

`Assert(...)` : Asserts that the given condition is true.

`abs(...)` : Computes the absolute value of a tensor.

`acos(...)` : Computes acos of x element-wise.

`acosh(...)` : Computes inverse hyperbolic cosine of x element-wise.

`add(...)` : Returns x + y element-wise.

`add_n(...)` : Adds all input tensors element-wise.

`argmax(...)` : Returns the index with the largest value across axes of a tensor.

`argmin(...)` : Returns the index with the smallest value across axes of a tensor.

`argsort(...)` : Returns the indices of a tensor that give its sorted order along an axis.

`as_dtype(...)` : Converts the given `type_value` to a `DType`.

`as_string(...)` : Converts each entry in the given tensor to strings.

`asin(...)` : Computes the trignometric inverse sine of x element-wise.

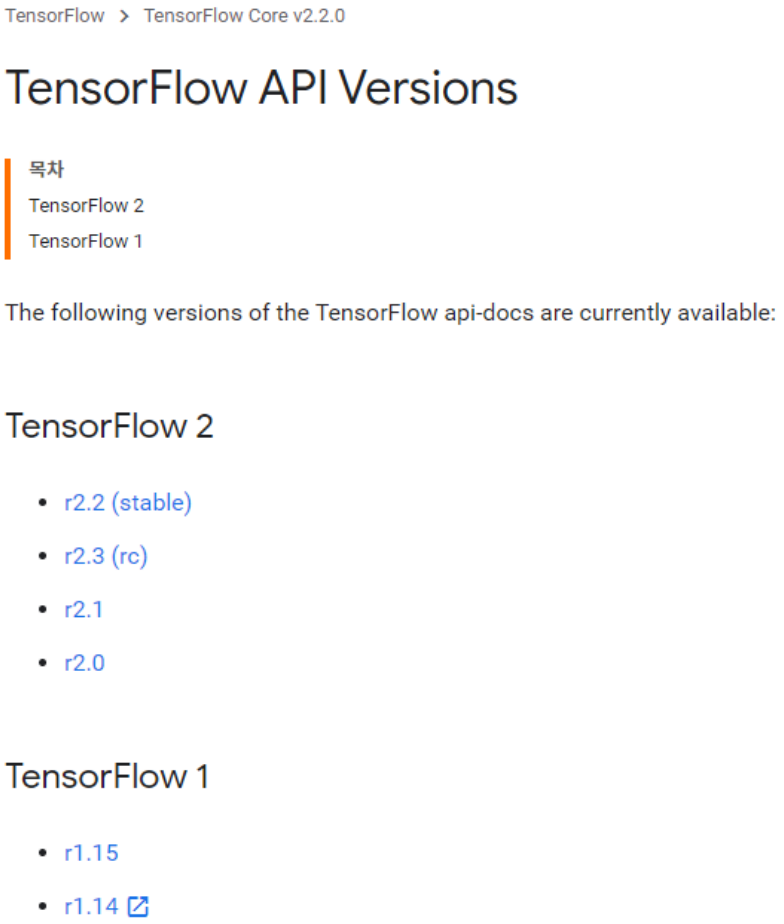
`asinh(...)` : Computes inverse hyperbolic sine of x element-wise.

`assert_equal(...)` : Assert the condition `x == y` holds element-wise.

`assert_greater(...)` : Assert the condition `x > y` holds element-wise.

버전

- https://www.tensorflow.org/versions



TensorFlow > TensorFlow Core v2.2.0

TensorFlow API Versions

목차

TensorFlow 2

TensorFlow 1

The following versions of the TensorFlow api-docs are currently available:

TensorFlow 2

- r2.2 (stable)
- r2.3 (rc)
- r2.1
- r2.0

TensorFlow 1

- r1.15
- r1.14

변수 Variables

- **딥러닝 학습에서 최적화 과정**
  - 모델의 매개변수(parameters) 즉, 가중치(및 편향)를 조정하는 것
- **변수(Variable) 사용**
  - 세션이 실행될 때 마다 그래프에서 고정된 상태를 유지하며, 계속 수정
- **변수 사용의 두 단계**
  - 변수 선언
    - **tf.Variable()함수를 사용해 변수를 만들고 어떤 값으로 초기화할지를 정의**
  - 초기화
    - **tf.global_variables_initializer() 메소드를 사용**
    - **세션에 초기화 연산을 수행해야 하며, 변수에 메모리를 할당하고 초기값을 설정하는 역할**
  - 다른 텐서 객체와 마찬가지로 변수 또한 그래프가 실행될 때 계산

플레이스홀더 placeholder

- **데이터를 입력 받는 비어있는 변수**
  - 입력 값을 넣어주기 위해 플레이스홀더(placeholder)
- **그래프가 실행되는 시점에 입력 데이터를 넣어주는데 사용**
  - 입력 데이터는 딕셔너리(dictionary)형태
  - session.run() 메소드에서 인자 feed_dict을 통해 전달
    - **딕셔너리의 키(key)는 플레이스홀더 변수 이름에 해당**
    - **값(value)은 list 또는 NumPy 배열**
  - sess.run(s, feed_dict={ph: data})
- **문제:**
  - a=7, $a^2 + b^2 = c^2$,

In [20]:
```
a = tf.placeholder(dtype=tf.float32, shape=None)
b = tf.placeholder(dtype=tf.float32, shape=None)

c = tf.sqrt(tf.pow(a,2) + tf.pow(b,2))

with tf.Session() as sess:
    result = sess.run(c, feed_dict={a: 7, b: 24})
    print(result)
```

```
25.0
```