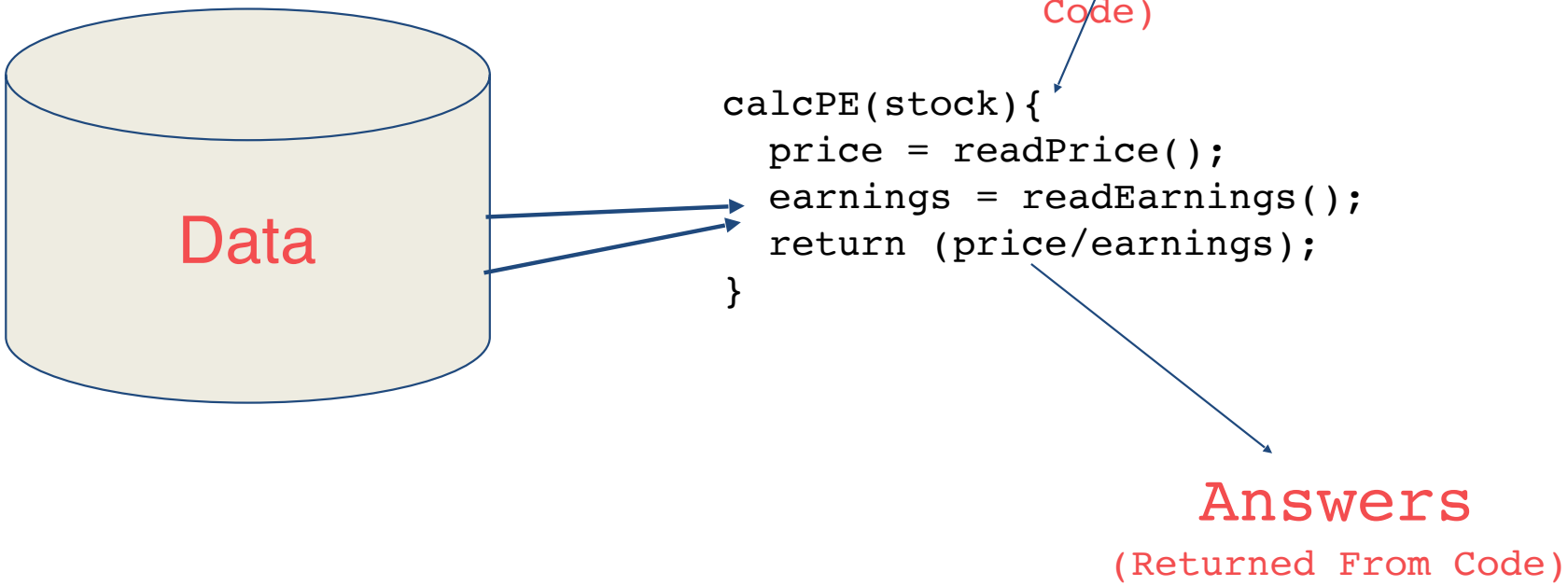
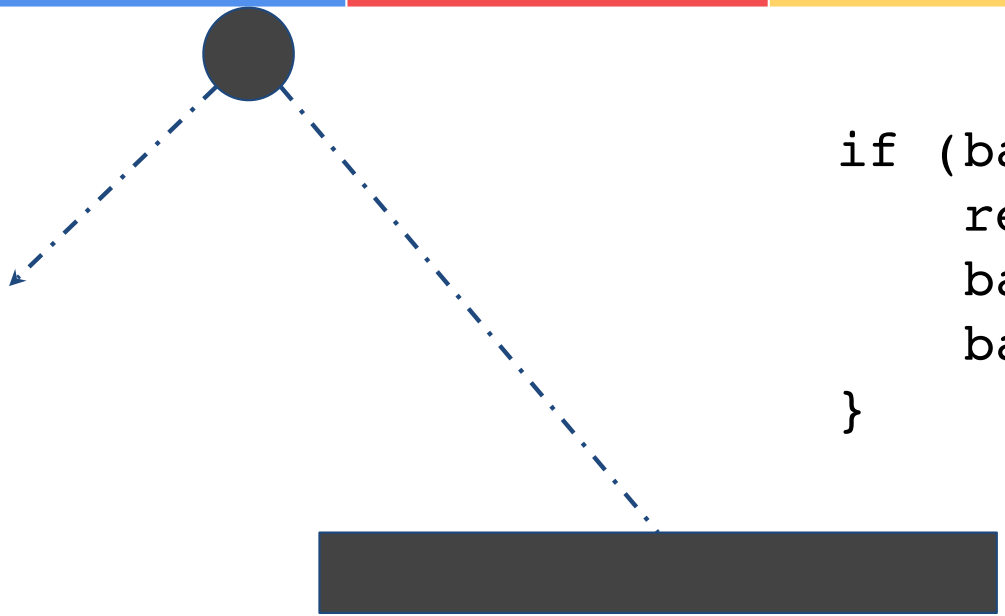
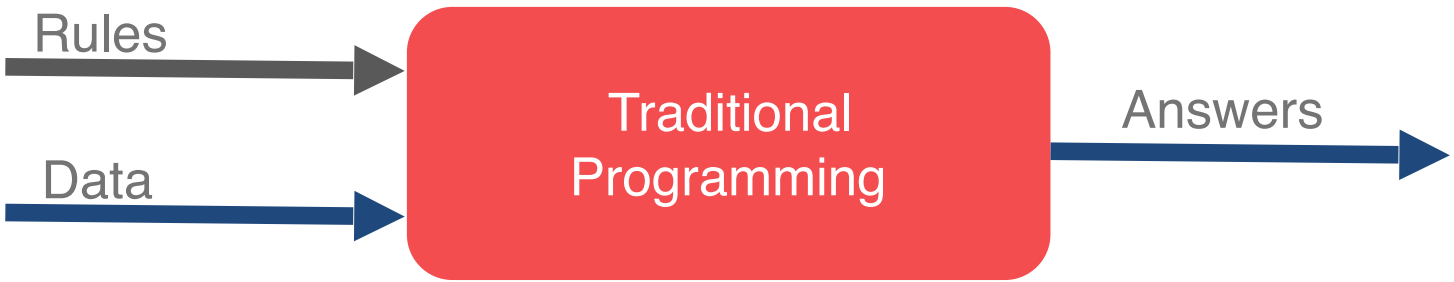
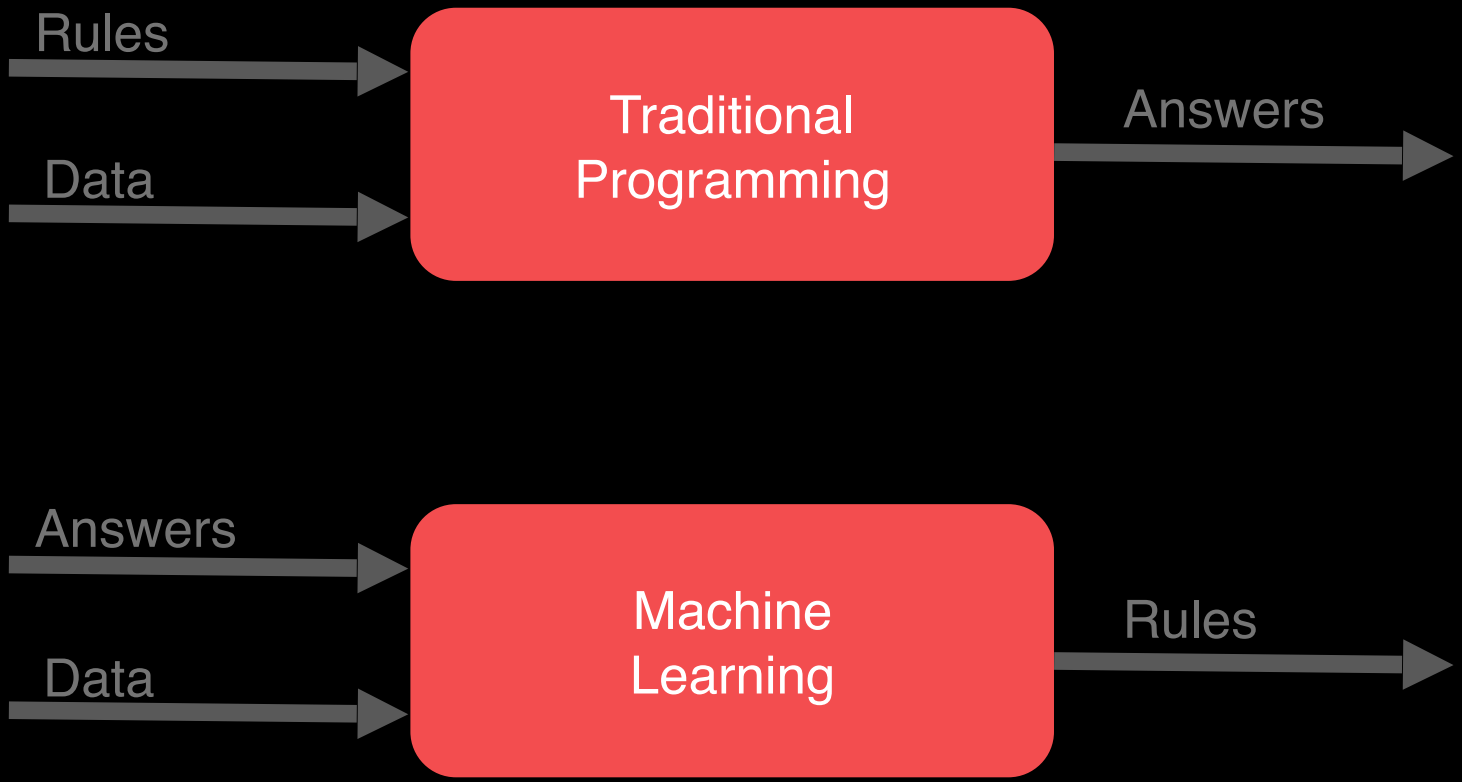
Copyright Notice


Rules
(Expressed in Code)
calcPE(stock){
price = readPrice();
earnings = readEarnings();
return (price/earnings);
}
Data
Answers
(Returned From Code)



```
if (ball.collide(brick)){
    removeBrick();
    ball.dx=-1*(ball.dx);
    ball.dy=-1*(ball.dy);
}
```


Rules
Data
Traditional
Programming
Answers


Rules
Data
Traditional Programming
Answers
Answers
Data
Machine Learning
Rules

Activity Recognition

```
if(speed<4){

status=WALKING;
}
```

Activity Recognition

```
if(speed<4){

status=WALKING;
}
```

```
if(speed<4){

status=WALKING;
} else {

status=RUNNING;
}
```

Activity Recognition

```
if(speed<4){

status=WALKING;
}
```

```
if(speed<4){

status=WALKING;
} else {

status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

Activity Recognition

```
if(speed<4){

status=WALKING;
}
```

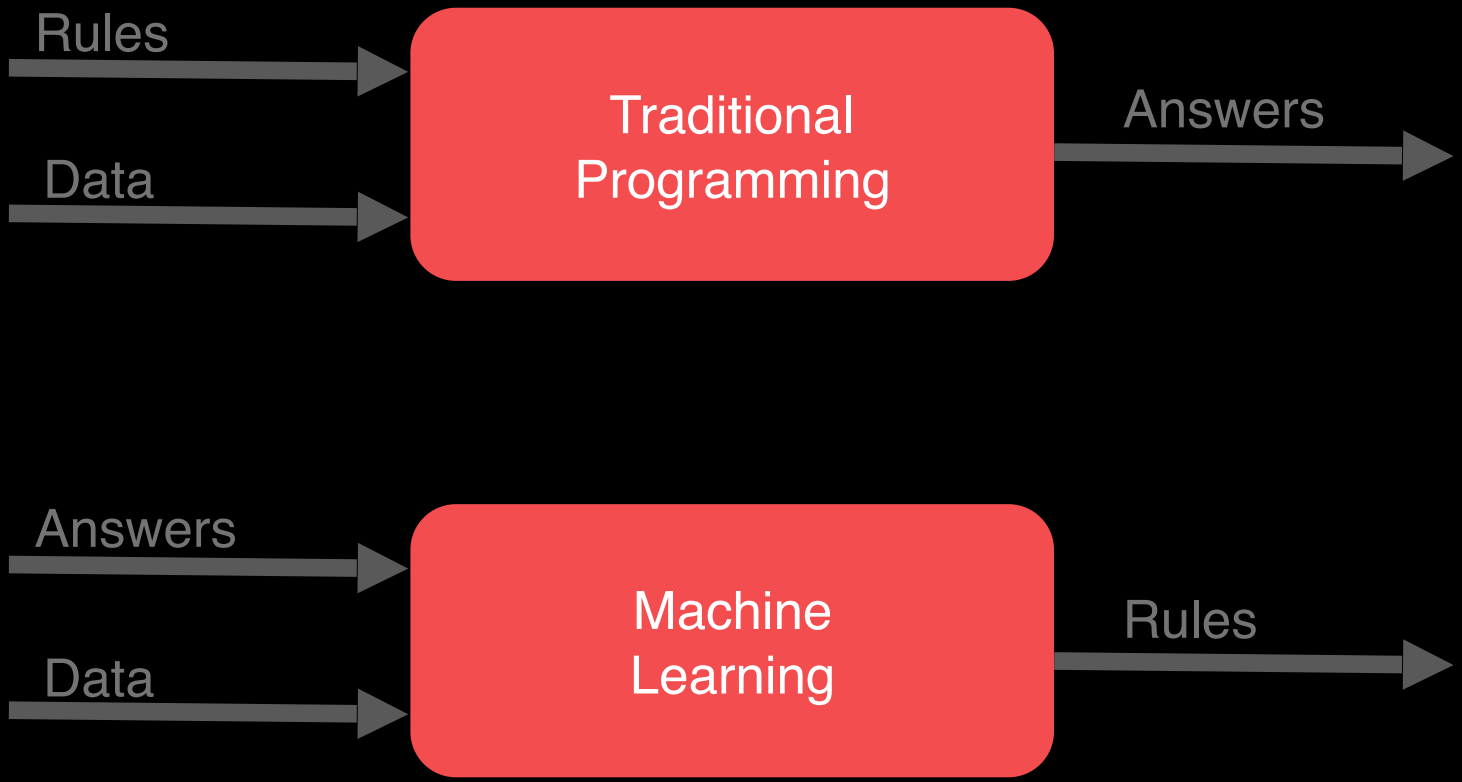
```
if(speed<4){

status=WALKING;
} else {

status=RUNNING;
}
```

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

```
// Oh crap
```


Rules
Data
Traditional Programming
Answers
Answers
Data
Machine Learning
Rules

Activity Recognition

```
0101001010100101
0101001010100101
1101010010100101
0101001010010101
00101010
```

Label = WALKING

```
1010100101001010
0101010100100100
0001001001111101
0101111010100100
11101011
```

Label = RUNNING

```
1001010011111010
1
0111010101110101
0
1110101010111101
0
1010111111100011
11010101
```

Label = BIKING

```
1111111110100111
0100111110101111
1
0101010111010101
0
1011101010101010
1
00111110
```

Label = GOLFING (Sort of)

X = -1, 0, 1, 2, 3, 4

Y = -3, -1, 1, 3, 5, 7

```
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

가장 간단한 단 1개의 뉴런만 가지고 있는 네트워크

Dense가 하나 → 하나의 레이어

Dense 1개 / Unit 1개 → Single Nueron

연결된 뉴런을 정의할 때

```
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')
```

평균제곱오차

또다른 추측을 제시

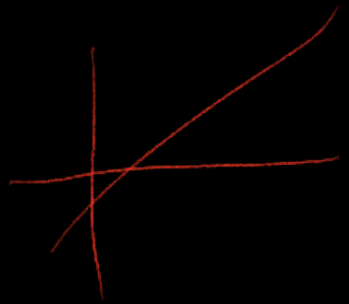
확률적 경사하강법

손실함수가 추측의 결과 판단, Optimizer에 전달

연속된 레이어들은 Sequence 안에 정의됨

```
xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
```

numpy



```
model.fit(xs, ys, epochs=500)
```

훈련 500번 반복

```
print(model.predict([10.0]))
```

예측 19에 가까운 값

```
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

```
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

```
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```