Theano tutorial part 1

AACIMP 2015
Sergii Gavrylov

Overview

- Introduction
- Symbolic variables
- Functions
- Shared variables / updates
- Gradients
- Substitution
- Random streams

What is Theano?

Université de Montréal

MILA

- Python library
- Computer algebra system
- Compiler

Where to find more info?

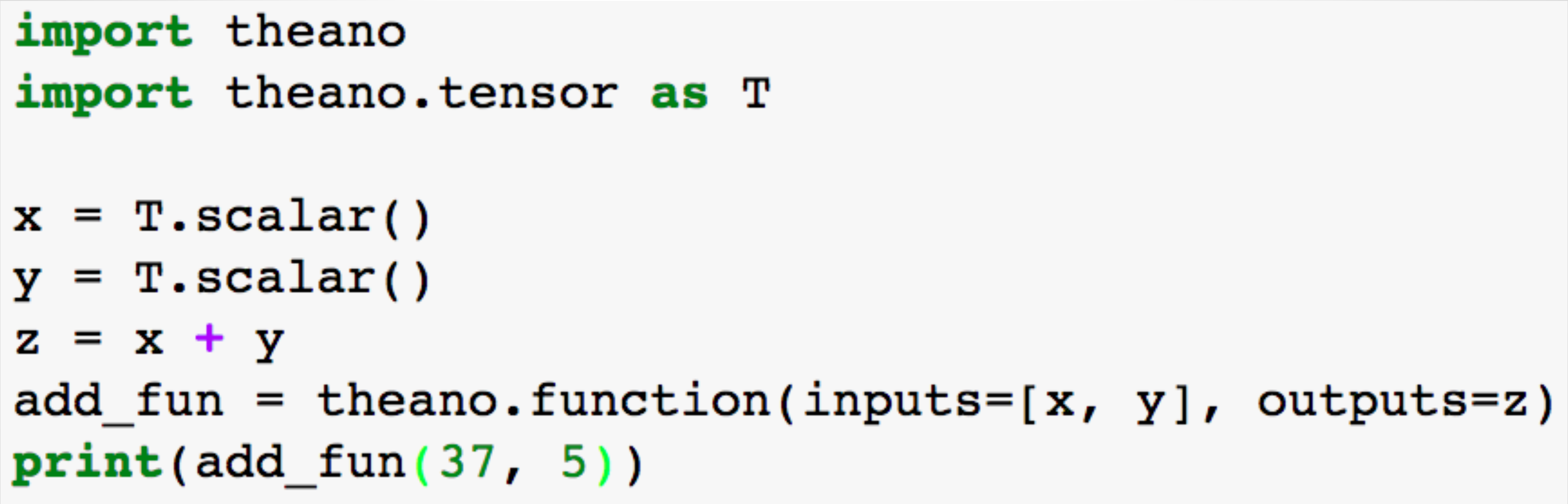
theano

- Website: deeplearning.net/software/theano
- User mailing list: groups.google.com/group/theano-users
- Deep learning tutorials: deeplearning.net/tutorial

Tutorial environment

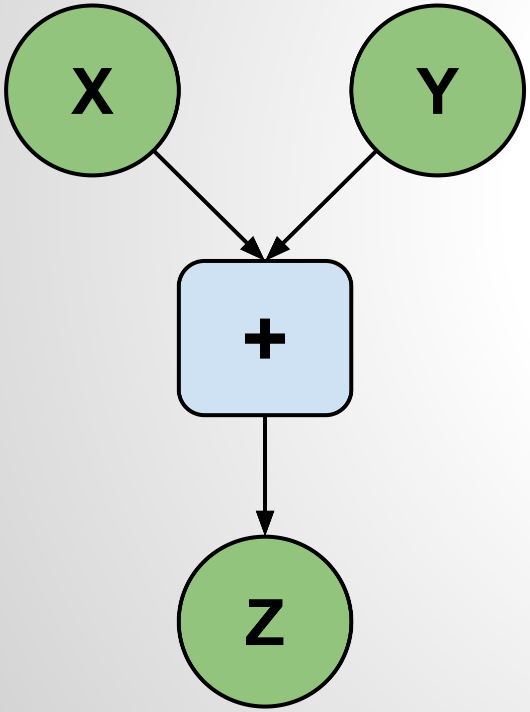
1. Go to --.--.---.---:---- to determine your workspace id that you will use through this tutorial
2. Go to --.--.---.---:---- to use your workspace
3. password: ----------

Scalar math

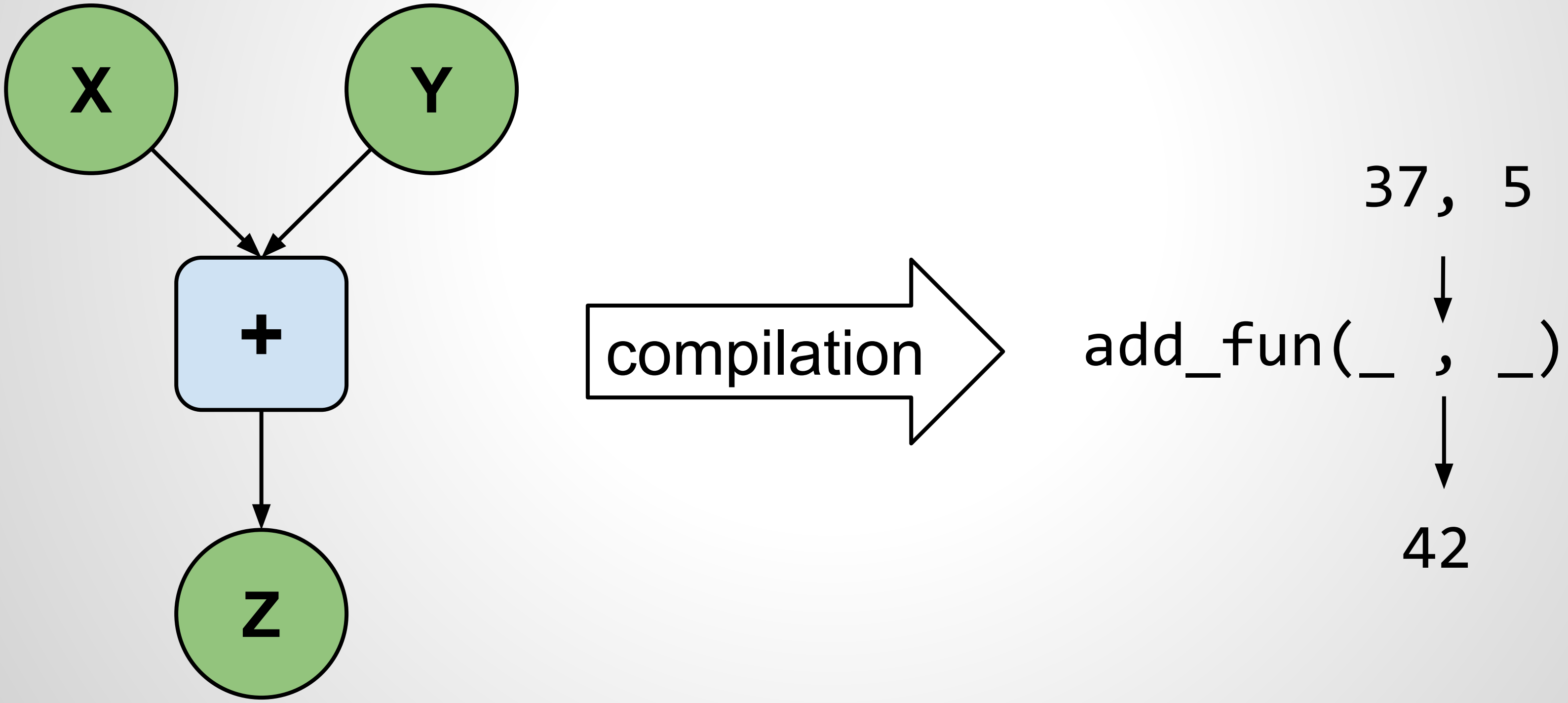


```
import theano
import theano.tensor as T

x = T.scalar()
y = T.scalar()
z = x + y
add_fun = theano.function(inputs=[x, y], outputs=z)
print(add_fun(37, 5))
```

Symbolic variables



Symbolic variables



compilation

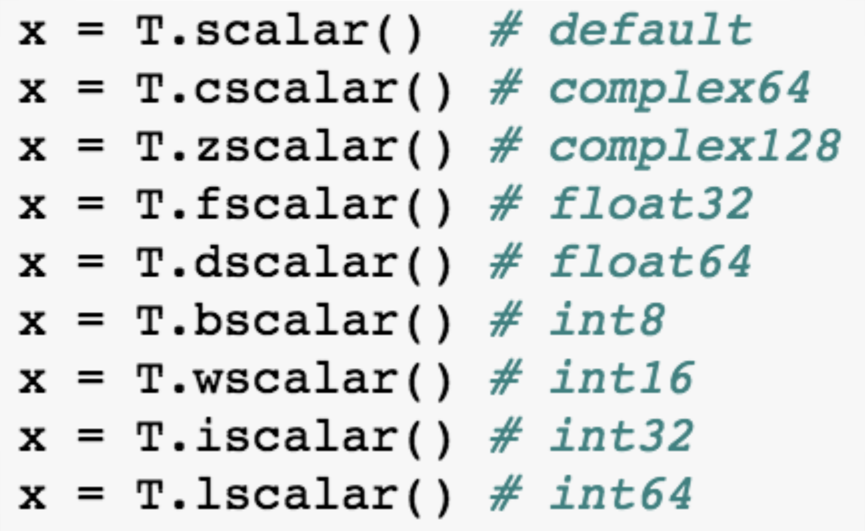
37, 5

add_fun(_ , _)

42

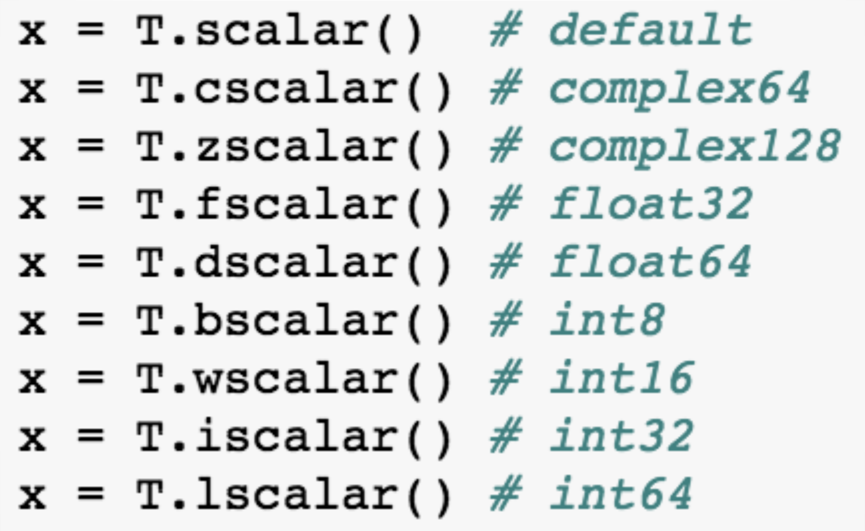
Task scalars

Data types

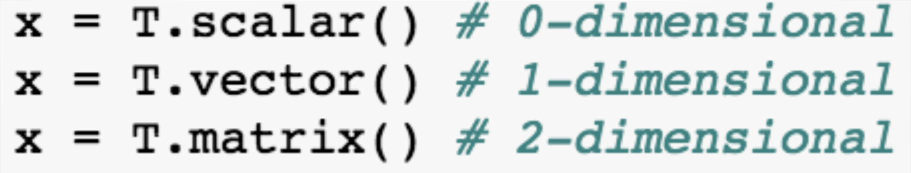


```
x = T.scalar()  # default
x = T.cscalar() # complex64
x = T.zscalar() # complex128
x = T.fscalar() # float32
x = T.dscalar() # float64
x = T.bscalar() # int8
x = T.wscalar() # int16
x = T.iscalar() # int32
x = T.lscalar() # int64
```

Data types

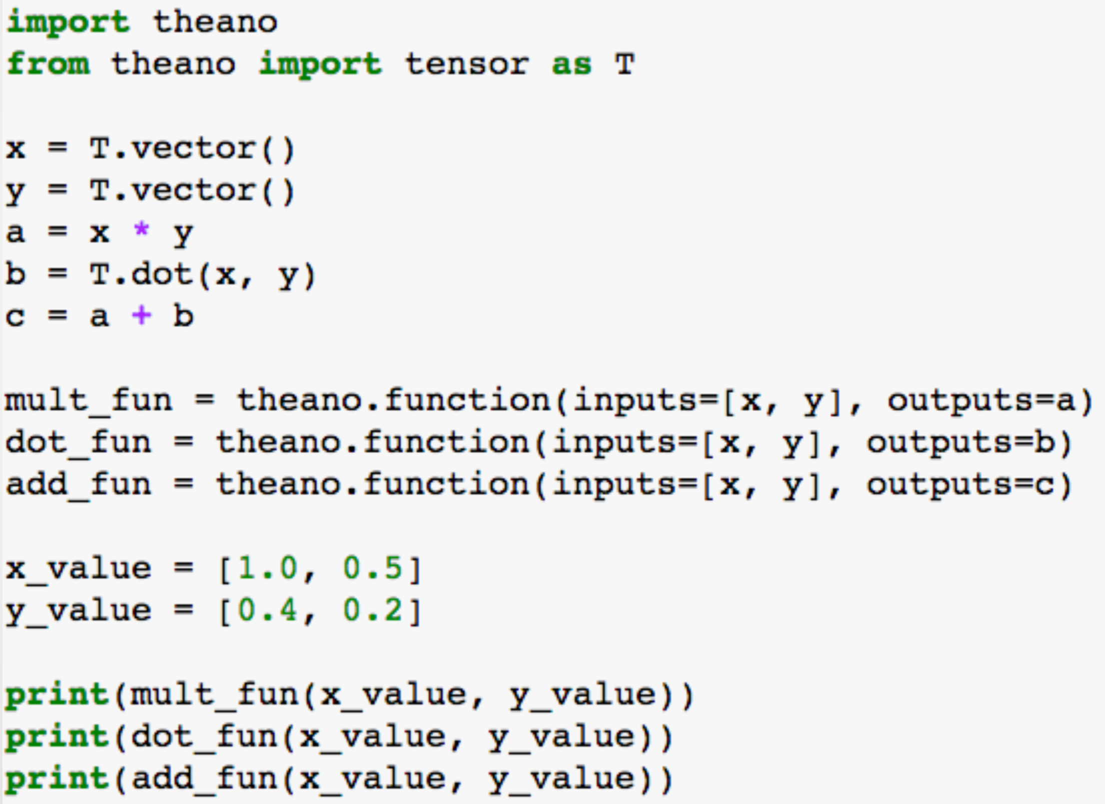


```
x = T.scalar()  # default
x = T.cscalar() # complex64
x = T.zscalar() # complex128
x = T.fscalar() # float32
x = T.dscalar() # float64
x = T.bscalar() # int8
x = T.wscalar() # int16
x = T.iscalar() # int32
x = T.lscalar() # int64
```



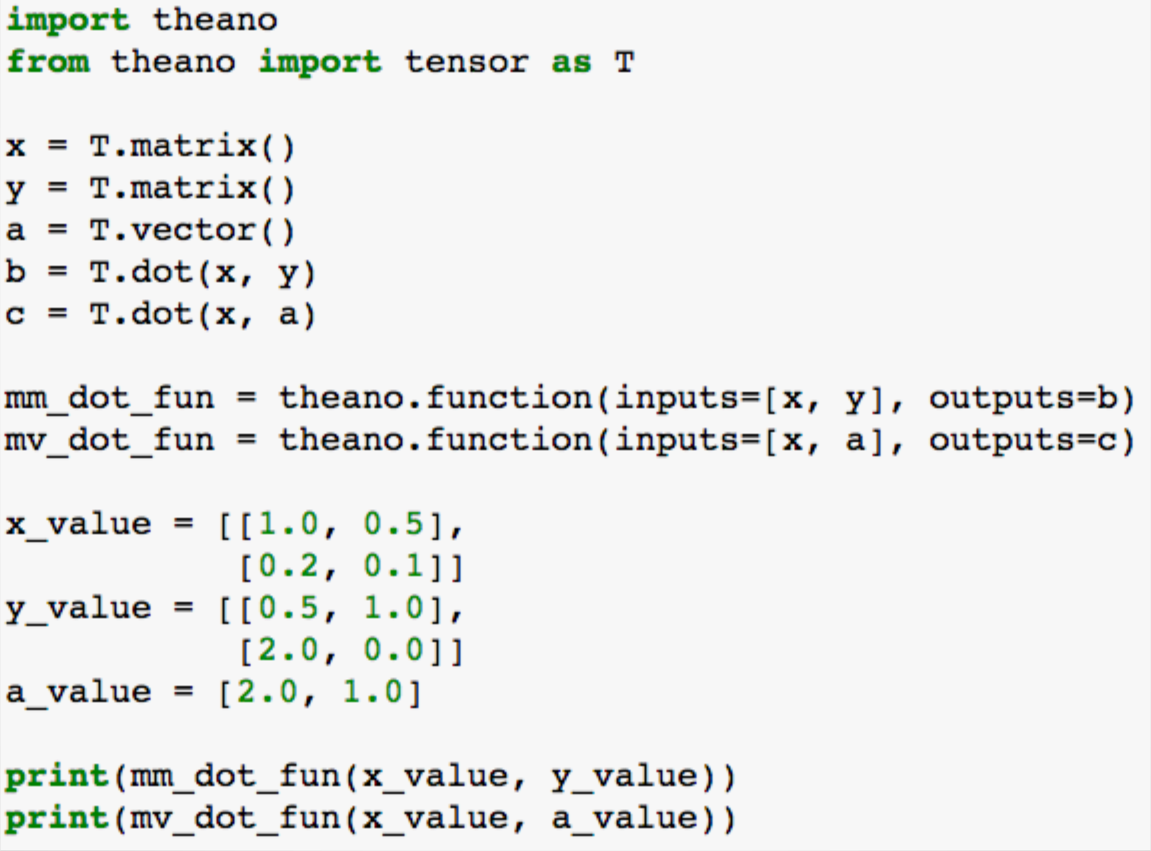
```
x = T.scalar() # 0-dimensional
x = T.vector() # 1-dimensional
x = T.matrix() # 2-dimensional
```

Vector math



```
import theano
from theano import tensor as T

x = T.vector()
y = T.vector()
a = x * y
b = T.dot(x, y)
c = a + b

mult_fun = theano.function(inputs=[x, y], outputs=a)
dot_fun = theano.function(inputs=[x, y], outputs=b)
add_fun = theano.function(inputs=[x, y], outputs=c)

x_value = [1.0, 0.5]
y_value = [0.4, 0.2]

print(mult_fun(x_value, y_value))
print(dot_fun(x_value, y_value))
print(add_fun(x_value, y_value))
```

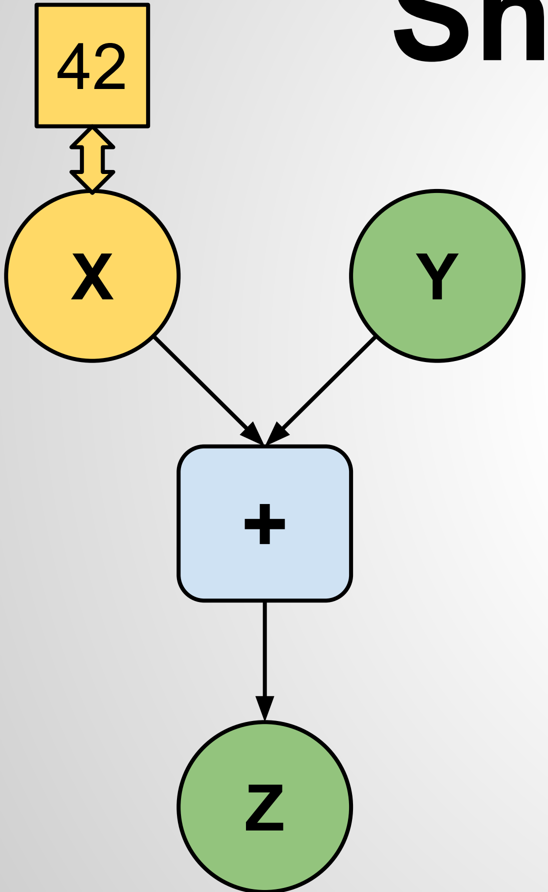
Matrix math



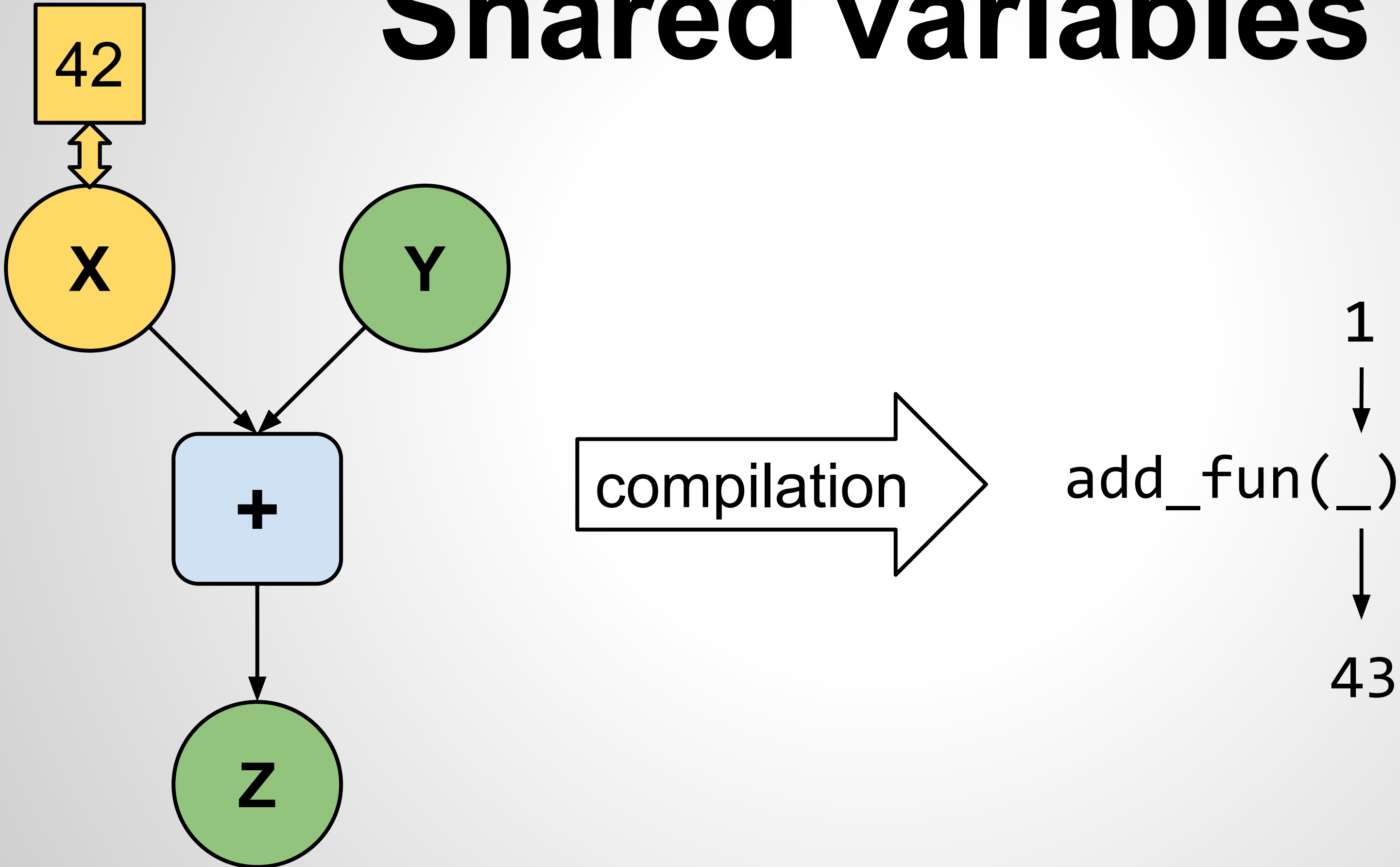
```python
import theano
from theano import tensor as T

x = T.matrix()
y = T.matrix()
a = T.vector()
b = T.dot(x, y)
c = T.dot(x, a)

mm_dot_fun = theano.function(inputs=[x, y], outputs=b)
mv_dot_fun = theano.function(inputs=[x, a], outputs=c)

x_value = [[1.0, 0.5],
           [0.2, 0.1]]
y_value = [[0.5, 1.0],
           [2.0, 0.0]]
a_value = [2.0, 1.0]

print(mm_dot_fun(x_value, y_value))
print(mv_dot_fun(x_value, a_value))
```

Task vectors_matrices

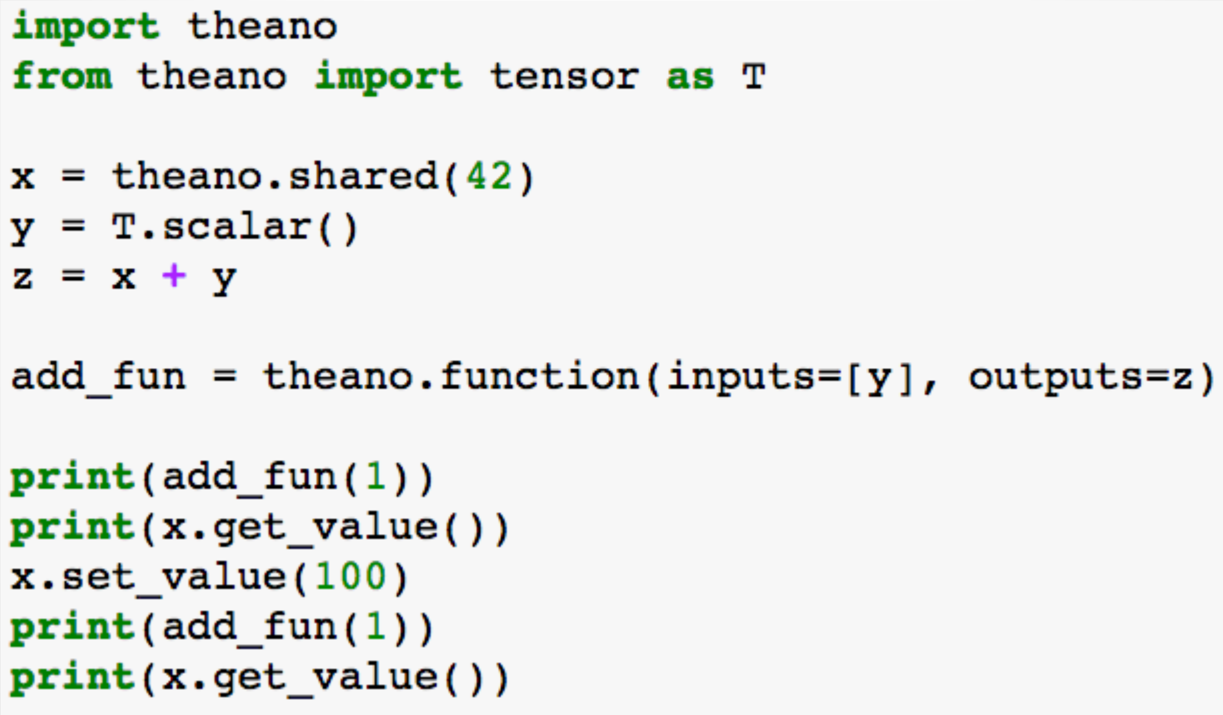
Shared variables



Shared variables

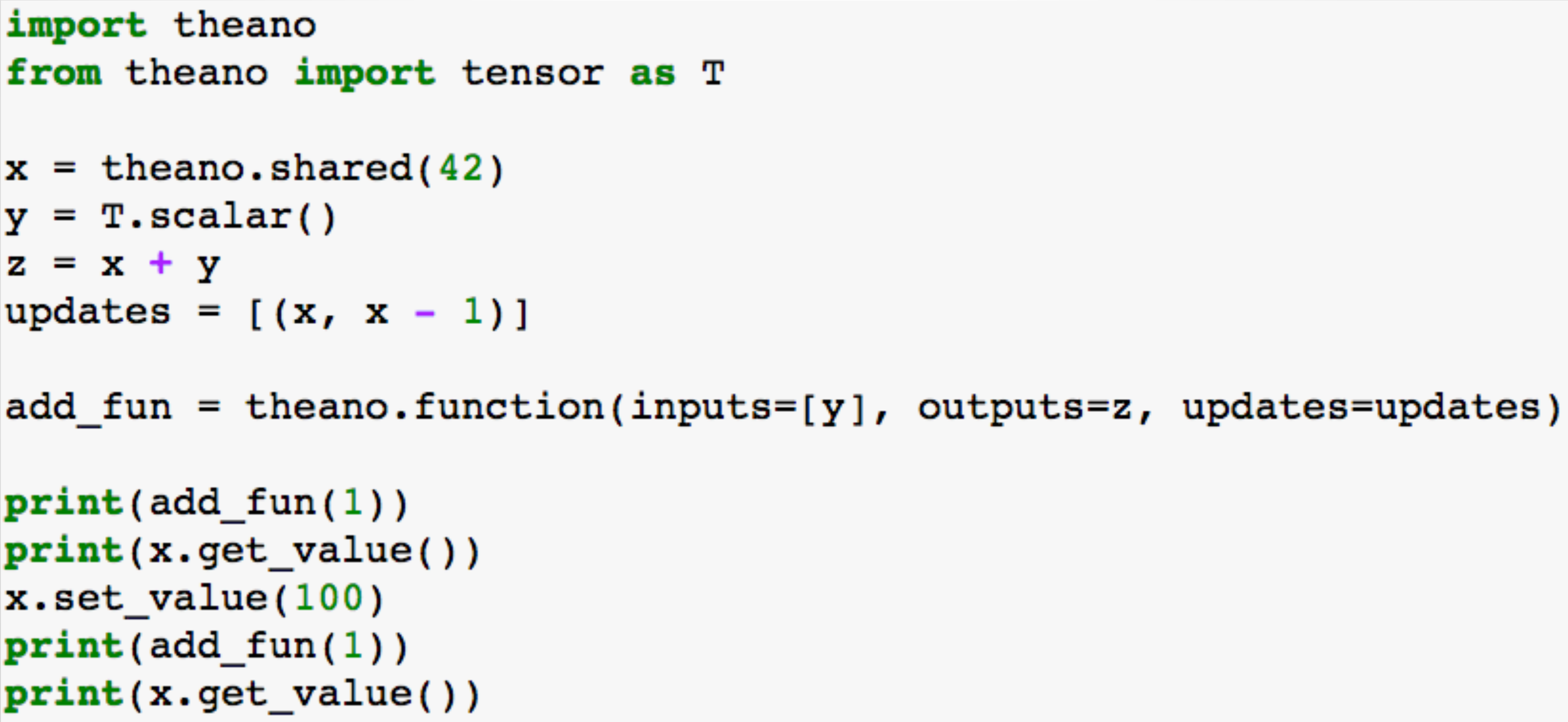


Shared variables



```python
import theano
from theano import tensor as T

x = theano.shared(42)
y = T.scalar()
z = x + y

add_fun = theano.function(inputs=[y], outputs=z)

print(add_fun(1))
print(x.get_value())
x.set_value(100)
print(add_fun(1))
print(x.get_value())
```

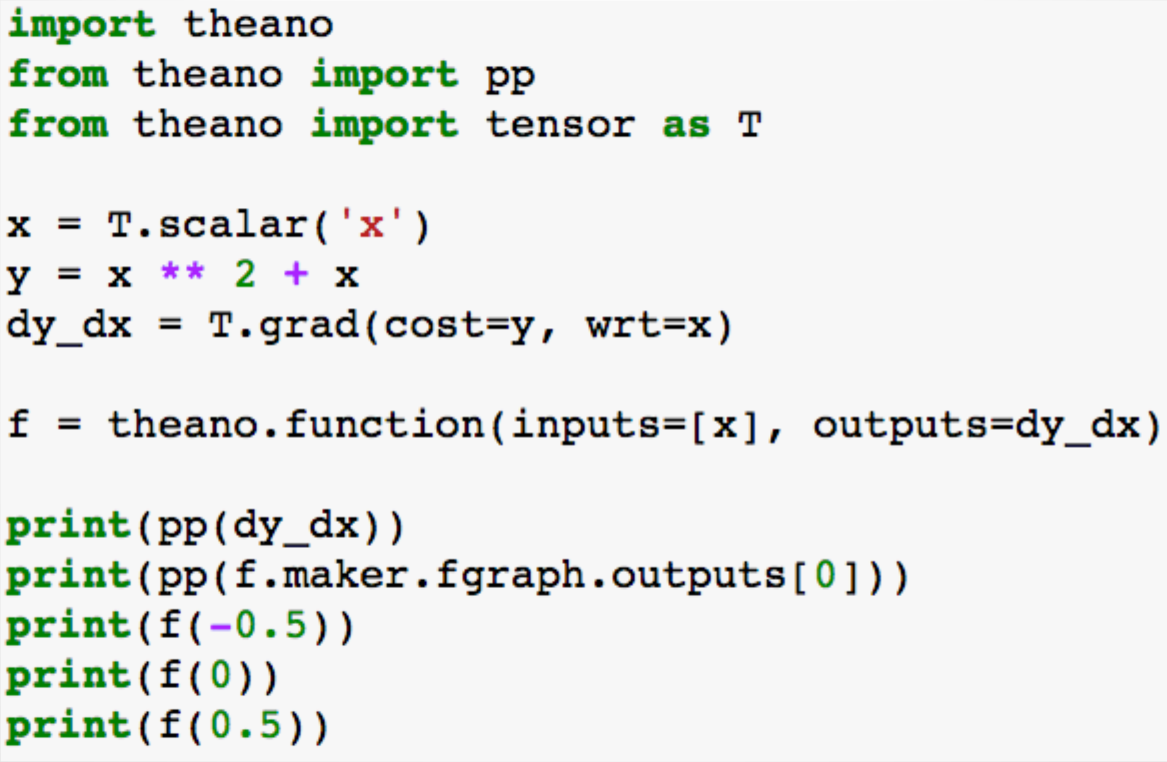
Updates



```
import theano
from theano import tensor as T

x = theano.shared(42)
y = T.scalar()
z = x + y
updates = [(x, x - 1)]

add_fun = theano.function(inputs=[y], outputs=z, updates=updates)

print(add_fun(1))
print(x.get_value())
x.set_value(100)
print(add_fun(1))
print(x.get_value())
```

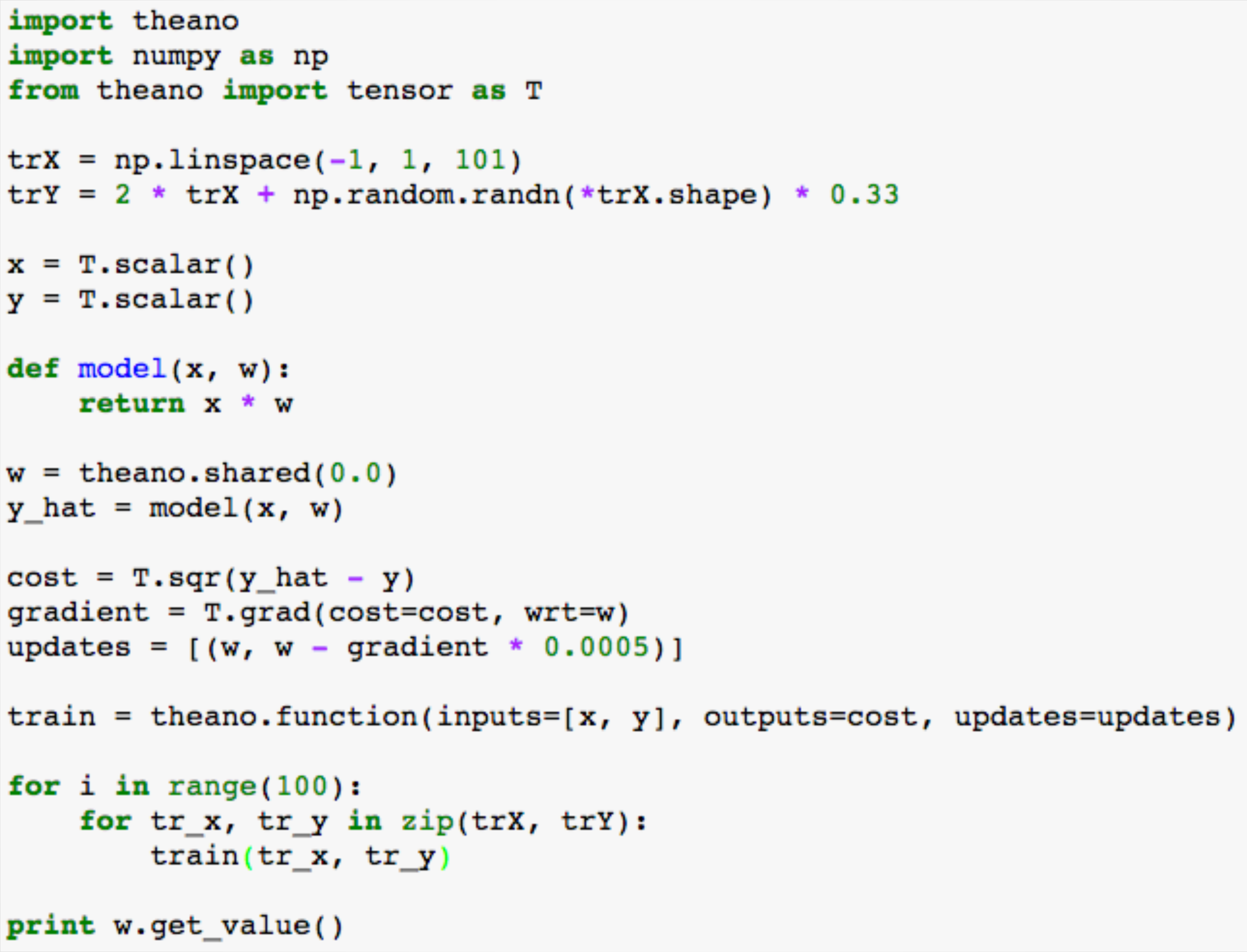
Task shared_updates

Symbolic differentiation



```
import theano
from theano import pp
from theano import tensor as T

x = T.scalar('x')
y = x ** 2 + x
dy_dx = T.grad(cost=y, wrt=x)

f = theano.function(inputs=[x], outputs=dy_dx)

print(pp(dy_dx))
print(pp(f.maker.fgraph.outputs[0]))
print(f(-0.5))
print(f(0))
print(f(0.5))
```

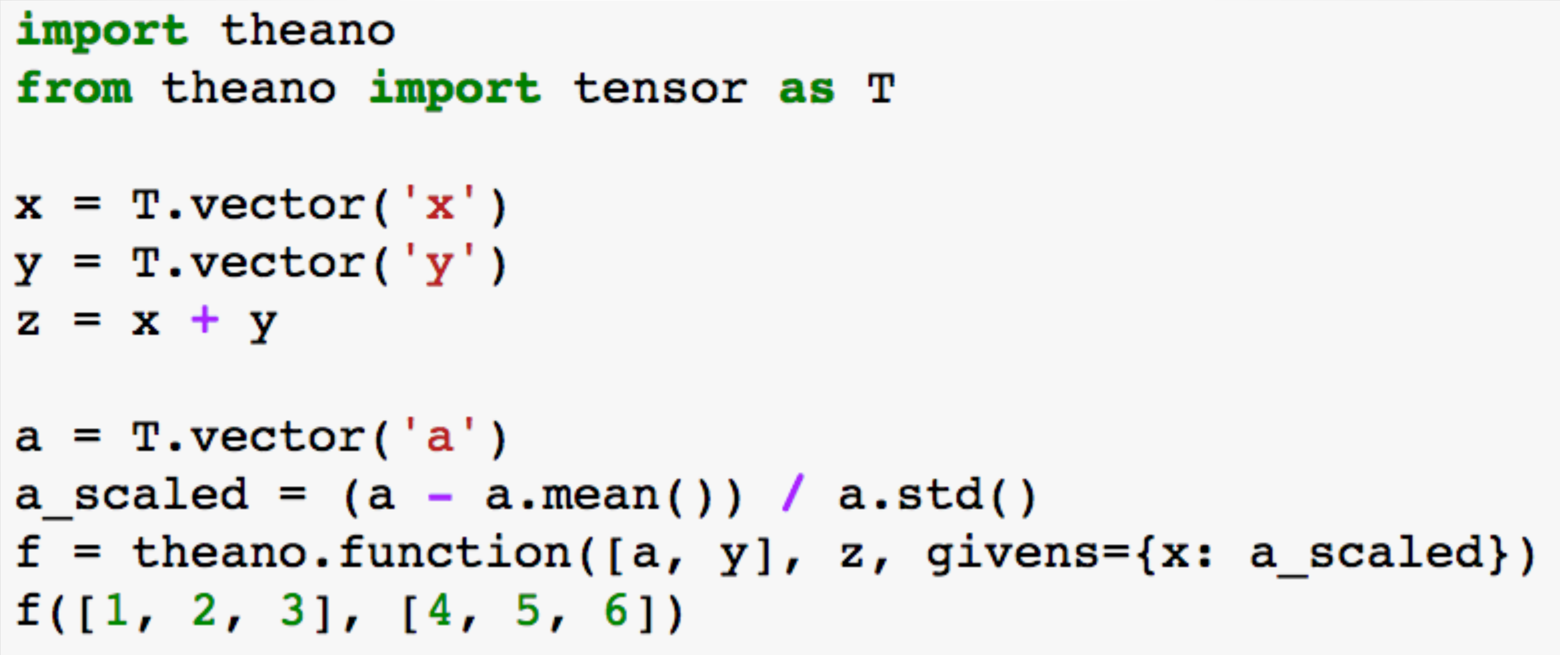
Task grad



```
import theano
import numpy as np
from theano import tensor as T

trX = np.linspace(-1, 1, 101)
trY = 2 * trX + np.random.randn(*trX.shape) * 0.33

x = T.scalar()
y = T.scalar()

def model(x, w):
    return x * w

w = theano.shared(0.0)
y_hat = model(x, w)

cost = T.sqr(y_hat - y)
gradient = T.grad(cost=cost, wrt=w)
updates = [(w, w - gradient * 0.0005)]

train = theano.function(inputs=[x, y], outputs=cost, updates=updates)

for i in range(100):
    for tr_x, tr_y in zip(trX, trY):
        train(tr_x, tr_y)

print w.get_value()
```

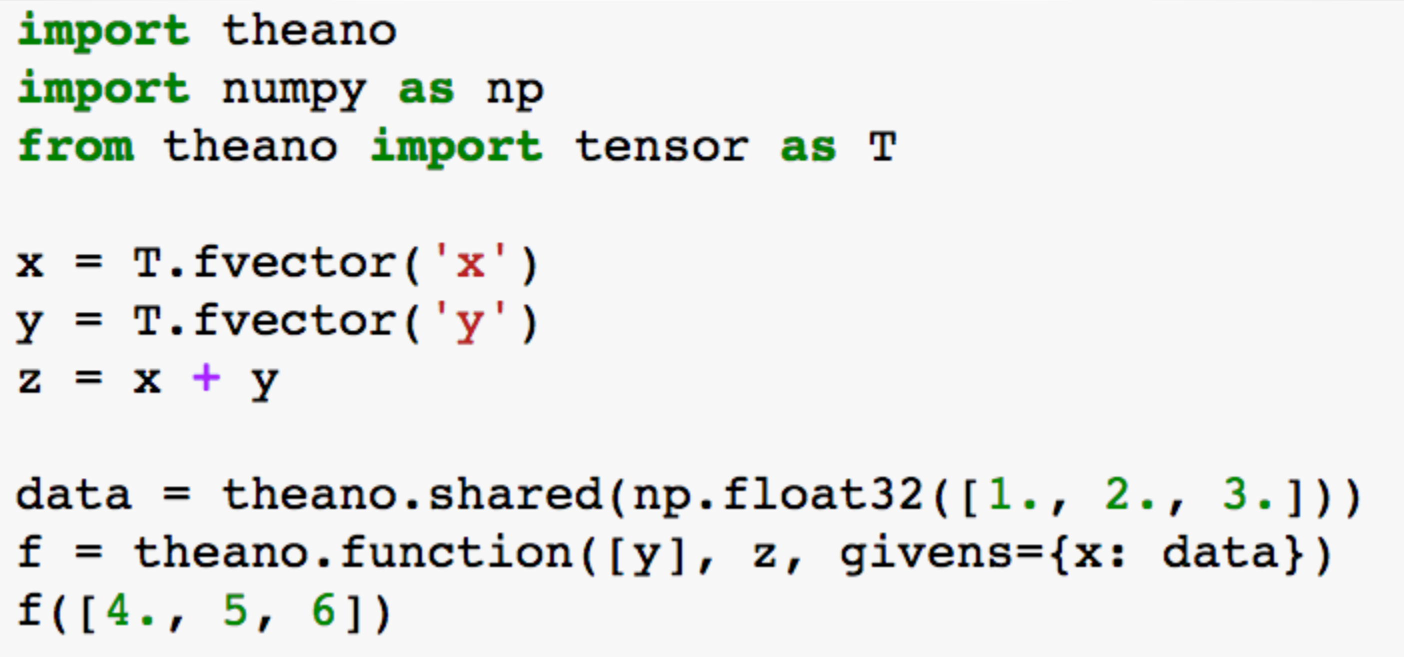
Task logistic_regression

Substitution



```
import theano
from theano import tensor as T

x = T.vector('x')
y = T.vector('y')
z = x + y

a = T.vector('a')
a_scaled = (a - a.mean()) / a.std()
f = theano.function([a, y], z, givens={x: a_scaled})
f([1, 2, 3], [4, 5, 6])
```

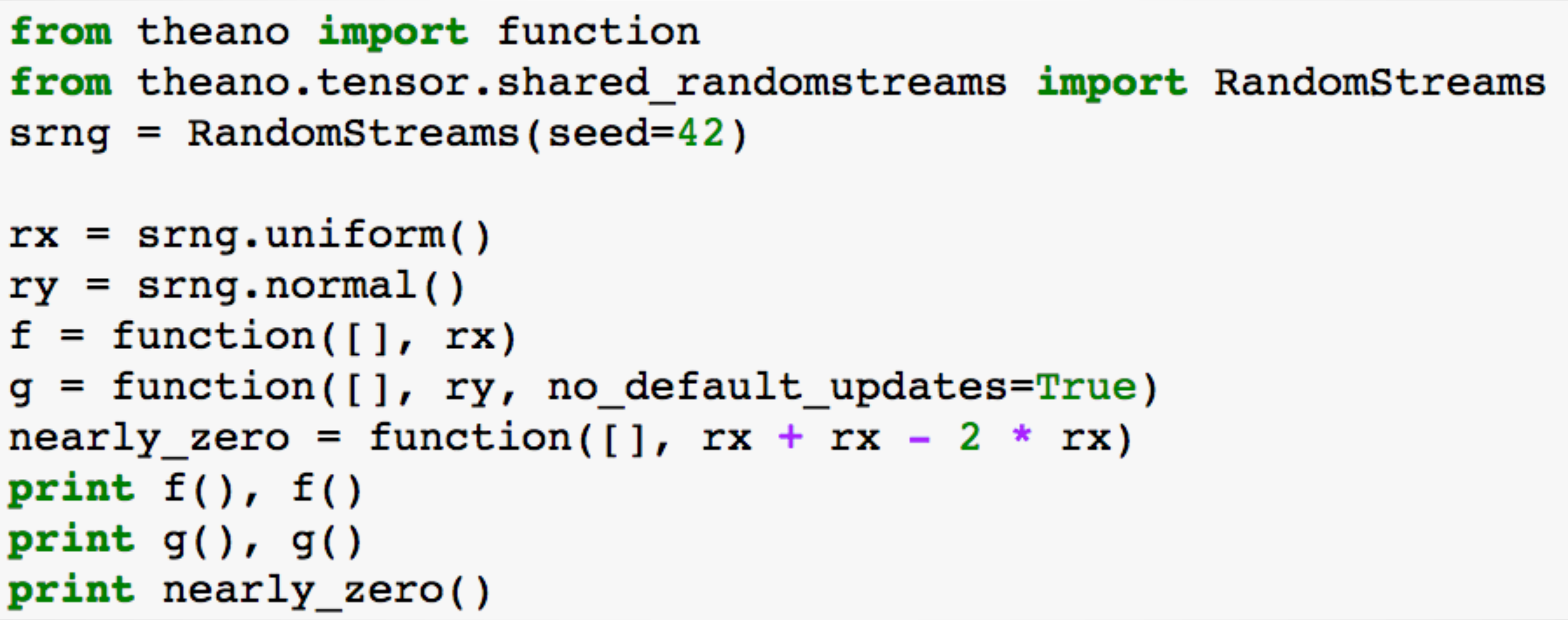
Substitution



```
import theano
import numpy as np
from theano import tensor as T

x = T.fvector('x')
y = T.fvector('y')
z = x + y

data = theano.shared(np.float32([1., 2., 3.]))
f = theano.function([y], z, givens={x: data})
f([4., 5, 6])
```

Task substitution

Random streams



```python
from theano import function
from theano.tensor.shared_randomstreams import RandomStreams
srng = RandomStreams(seed=42)

rx = srng.uniform()
ry = srng.normal()
f = function([], rx)
g = function([], ry, no_default_updates=True)
nearly_zero = function([], rx + rx - 2 * rx)
print f(), f()
print g(), g()
print nearly_zero()
```

Task random_stream

Conclusion

- Theano is a tool that combines the best of both worlds: it is easy to code and fast to execute!
- It is used in both academia and industry.
- Symbolic paradigm is extremely flexible, but it might be not suitable for everyone.
- Part 2 is coming.