Debug operator

Currently, in order to debug one has to use the corresponding numpy functions
For eg

```
import mxnet as mx
import numpy as np
a = mx.nd.zeros(1)
print(a)
print(np.isinf(a.asnumpy()))
b = mx.nd.array(np.inf)
print(b)
print(np.isinf(b.asnumpy()))
```

Output

```
[0.]
<NDArray 1 @cpu(0)>
[False]

inf
<NDArray  @cpu(0)>
True
```

Ideal scenario,

```
import mxnet as mx
a = mx.nd.zeros(1)
print(a)
print(mx.nd.isinf(a))
```

```
[0.]
<NDArray 1 @cpu(0)>
[False]
```

test_ndarray.py `mx.nd.zeros(shape)` → (anaconda/site-packages/mxnet/ndarray/utils.py : `zeros_ndarray()` → mxnet/ndarray/ndarray.py : `zeros()` → gen_internal.py `_internal._zeros`

Unable to trace the actual implementation of zeros
Lin mentioned - /src/operator/tensor/init_op.cc (how did he find it?)

init_op.cc - zeros (operator registration)
init_op.h - zeros (op fn definition)

is_inf (tensorflow adopts from numpy)
Hence cloned the numpy repo to understand how is_inf is made

BLAS - basic linear algebra programs (OpenBLAS - optimized)
LAPACK
ATLAS - automatically tuned linear algebra software (once a SOTA, now inferior since

originally GotoBlas → Kazushige Goto, John Henry of information age.

> legend in the supercomputing community because of his solitary crusade. research associate at the Texas Advanced Computing Center at the University of Texas at Austin. IBM machines that overtook several supercomputers used Goto's BLAS. used by 4 of the world's 11 fastest computers.

Atlas - Jack Dongarra, a computer scientist at the University of Tennessee
automated effort to find the most efficient way to solve linear algebra functions for specific microprocessors compared to meticulous code written by hand by Goto

While going through numpy code

- RST - c-api.coremath.rst (reStructured Text)
- Finding np.array()
    a. It's not a class, but a “convenience function”
    b. It is aliased to multi-array (C code) .so = shared object (compiled code)