IPA 주관 인공지능센터 기본(fundamental) 과정

- **GitHub link: here**
- **E-Mail: windkyle7@gmail.com**

NumPy Part 2

ufunc

- 벡터화된 연산
  - **NumPy**는 이미 컴파일된 코드로 만들어진 다양한 벡터화된 연산을 제공
  - 이런 종류의 연산을 **ufunc**라고 함
  - 더 간결하고 효율적
  - 원하는 것을 설명하는 편이 명령을 내리는 것보다 낫다 **(**데이터 타입을 사용하자**)**
  - 이를 통해 컴파일된 언어의 성능을 끌어다 쓸수 있음
  - 벡터화한 것이 명시적인 루프보다 좋다
  - 모든 연산은 기본적으로 개별원소마다**(elementwise)**적용
  - 파이썬에서 기본적으로 제공하는 함수와 섞어 쓰지 않을것
- **np.info()**

- 함수와 메소드를 동일하게 처리
  - **numpy** 모듈에 함수**, ndarray** 내의 메소드가 이 중으로 지원하는 함수와 메소드가 많지만 용도에 맞춰사용 **->** 메소드 사용을 권고

In [0]:

```
def add(self, x, y):
    return x + y


class A(object):
    add = add
```

In [0]:

```
a = A()
```

In [0]:

```
a.add(5, 5)
```

Out[0]:

```
10
```

In [0]:

```
type(a.add)
```

Out[0]:

```
method
```

In [0]:

```
type(add)
```

Out[0]:

```
function
```

- **1**개의 배열에 대한 **ufunc** 함수**: Unary univerasal functions**
- **2**개의 배열 간 **ufunc** 함수**: Binary universal functions**

Unary ufuncs

Function	Description
abs, fabs	Compute the absolute value element-wise for integer, floating point, or comple x values. Use fabs as a faster alternative for non-complex-valued data
sqrt	Compute the square root of each element. Equivalent to arr ** 0.5
square	Compute the square of each element. Equivalent to arr ** 2
exp	Compute the exponent ex of each element
log, log10, log2, log1p	Natural logarithm (base e), log base 10, log base 2, and log(1 + x), respectively
sign	Compute the sign of each element: 1 (positive), 0 (zero), or -1 (negative)
ceil	Compute the ceiling of each element, i.e. the smallest integer greater than or e qual to each element
floor	Compute the floor of each element, i.e. the largest integer less than or equal t o each element
rint	Round elements to the nearest integer, preserving thedtype
modf	Return fractional and integral parts of array as separate array
isnan	Return boolean array indicating whether each value isNaN (Not a Number)
isfinite, isinf	Return boolean array indicating whether each element is finite (non-inf, non-Na N) or infinite, respectively
cos, cosh, sin, sinh, tan, t anh	Regular and hyperbolic trigonometric functions
arccos, arccosh, arcsin, ar csinh, arctan, arctanh	Inverse trigonometric functions

logical_not	Compute truth value of not x element-wise. Equivalent to -arr.

Binary universal functions

Function	Description
add	Add corresponding elements in arrays
subtract	Subtract elements in second array from first array
multiply	Multiply array elements
divide, floor_divide	Divide or floor divide (truncating the remainder)
power	Raise elements in first array to powers indicated in second array
maximum, fmax	Element-wise maximum. fmax ignores NaN
minimum, fmin	Element-wise minimum. fmin ignores NaN
mod	Element-wise modulus (remainder of division)
copysign	Copy sign of values in second argument to values in first argum ent
greater, greater_equal, less, less_equal, equal, not_equal	Perform element-wise comparison, yielding boolean array. Equiva lent to infix operators >, >=, <, <=, ==, !=
logical_and, logical_or, logical_xor	Compute element-wise truth value of logical operation. Equivale nt to infix operators & |, ^

Basic array statistical methods

Method	Description
sum	Sum of all the elements in the array or along an axis. Zero-length arrays have sum 0.
mean	Arithmetic mean. Zero-length arrays have NaN mean.
std, var	Standard deviation and variance, respectively, with optional degrees of fre edom adjustment (default denominator n).
min, max	Minimum and maximum.
argmin, argmax	Indices of minimum and maximum elements, respectively.
cumsum	Cumulative sum of elements starting from 0
cumprod	Cumulative product of elements starting from 1

Array set operations

Method	Description
unique(x)	Compute the sorted, unique elements in x
intersect1d(x, y)	Compute the sorted, common elements in x and y
union1d(x, y)	Compute the sorted union of elements
in1d(x, y)	Compute a boolean array indicating whether each element of x is contained in y
setdiff1d(x, y)	Set difference, elements in x that are not in y
setxor1d(x, y)	Set symmetric differences; elements that are in either of the arrays, but not both

In [0]:

```
import numpy as np
```

indexing & slicing

In [0]:

```
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [0]:

```
x
```

Out[0]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

`i:j:k` 여기서 `i`는 시작 인덱스이고, `j`는 정지 인덱스, `k`는 단계이다.

In [0]:

```
x[1:7:2]
```

Out[0]:

```
array([1, 3, 5])
```

음수 `i`와 `j`는 `n + i`와 `n + j`로 해석된다. 여기서 `n`은 해당 차원의 요소 수이다. 음수 `k`는 더 작은 인덱스로 이동한다.

In [0]:

```
x[-2:10]
```

Out[0]:

```
array([8, 9])
```

In [0]:

```
x[-3:3:-1]
```

Out[0]:

```
array([7, 6, 5, 4])
```

In [0]:

```
x[5:]
```

Out[0]:

```
array([5, 6, 7, 8, 9])
```

In [0]:

```
x = np.array([[[1], [2], [3]], [[4], [5], [6]]])
```

In [0]:

```
x
```

Out[0]:

```
array([[[1],
        [2],
        [3]],

       [[4],
        [5],
        [6]]])
```

In [0]:

```
x.shape
```

Out[0]:

```
(2, 3, 1)
```

Ellipsis

In [0]:

```
x[..., 0]
```

Out[0]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

np.newaxis

`None`에 대한 **numpy** 별칭으로 배열을 인덱싱하는 데 유용하다.

In [0]:

```
np.newaxis is None
```

Out[0]:

```
True
```

In [0]:

```
x[:, np.newaxis, :, :].shape
```

Out[0]:

```
(2, 1, 3, 1)
```

Integer array indexing

In [0]:

```
x = np.array([[1, 2], [3, 4], [5, 6]])
```

In [0]:

```
x[[0, 1, 2], [0, 1, 0]]
```

Out[0]:

```
array([1, 4, 5])
```

In [0]:

```
x = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]])
```

In [0]:

```
rows = np.array([[0, 0],
                 [3, 3]], dtype=np.intp)
```

In [0]:

```
columns = np.array([[0, 2],
                    [0, 2]], dtype=np.intp)
```

In [0]:

```
x[rows, columns]
```

Out[0]:

```
array([[ 0,  2],
       [ 9, 11]])
```

np.ix_

다차원 배열을 인덱싱한다.

In [0]:

```
a = np.arange(10).reshape(2, 5)
```

In [0]:

In [0]:

```
a
```

Out[0]:

```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

In [0]:

```
ixgrid = np.ix_([0, 1], [2, 4])
```

In [0]:

```
ixgrid
```

Out[0]:

```
(array([[0],
        [1]]), array([[2, 4]]))
```

In [0]:

```
ixgrid[0].shape, ixgrid[1].shape
```

Out[0]:

```
((2, 1), (1, 2))
```

In [0]:

```
a[ixgrid]
```

Out[0]:

```
array([[2, 4],
       [7, 9]])
```

In [0]:

```
a[[0, 1]][:, [2,3]]
```

Out[0]:

```
array([[2, 3],
       [7, 8]])
```