Python for Scientific Data Analysis

NumPy/SciPy

Section 2: Array Arithmetic and Universal Functions

Arithmetic

As stated in the previous section, arrays are important because they enable you to express batch operations on data without writing any for loops. I.e. they allow for *vectorization* which will almost always be faster than looping. Any arithmetic operations between equal-size arrays applies the operation element-wise.

Beyond the simple example given before, here are some examples of array arithmetic:

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

arr

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

arr * arr

#array([[ 1., 4., 9.],
#[ 16., 25., 36.]])

arr - arr
#array([[ 0., 0., 0.],
#[ 0., 0., 0.]])
```

Arithmetic operations with scalars propagate the scalar argument to each element in the array:

```
1 / arr

#array([[ 1. , 0.5 , 0.3333],
#[ 0.25 , 0.2 , 0.1667]])

arr ** 0.5

#array([[ 1. , 1.4142, 1.7321],
#[ 2. , 2.2361, 2.4495]])
```

Comparisons between arrays of the same size yield boolean arrays:

```
arr2 = np.array([[0., 4., 1.], [7., 2., 12.]])
arr2

arr2 > arr
#array([[False, True, False],
#[ True, False, True]], dtype=bool)
```

Operations between arrays of different sizes is called *broadcasting*: we will discuss that later.

Universal Functions

NumPy contains *universal functions* that perform vectorized operaitons on NumPy arrays.

unary universal functions perform element-wise transformations and are called as `np.functionname(array)`. E.g. `np.exp(array)`, `np.sqrt(array)`, `np.floor(array)`. Here are some examples:

Function	Description
abs, fabs	Compute the absolute value element-wise for integer, floating-point, or complex values
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., smallest integer greater than or equal to that number)
floor	Compute the floor of each element (i.e., the largest integer less than or equal to each element)
rint	Round elements to the nearest integer, preserving the dtype
modf	Return fractional and integral parts of array as a separate array
isnan	Return boolean array indicating whether each value is NaN (Not a Number)
isfinite, isinf	Return boolean array indicating whether whether each element is finite (non-inf, non-NaN) or infinite,respectively
cos, cosh, sin,sinh, tan, tanh	Regular and hyperbolic trigonometric functions
arccos, arccosh,arcsin, arcsinh,arctan, arctanh	Inverse trigonometric functions
logical_not	Compute truth value of not x element-wise (equivalent to ~arr).

binary universal functions two two arrays and return a single array as a result. They are called as `np.functionname(array1,array2)`. Here are some examples:

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 argument
greater, greater_equal,less, less_equal,equal, not_equal	Perform element-wise comparison, yielding boolean array (equivalent to infix operators >, >=, <, <=, ==, !=)
logical_and,logical_or, logical_xor	Compute element-wise truth value of logical operation (equivalent to infix operators &

Some ufuncs can return multiple arrays -- e.g. `np.modf` which returns the fractional and integral parts of a floating point array. But these are uncommon.