Numerical Python

Volodymyr Byelobrov

PyConPL 2019

September 15, 2019

Table of Contents

Guido van Rossum first intentions were to create a script language suitable for teaching purposes. Still because of his interest in Maths, a lot of mathematical features were implemented in the core of Python.

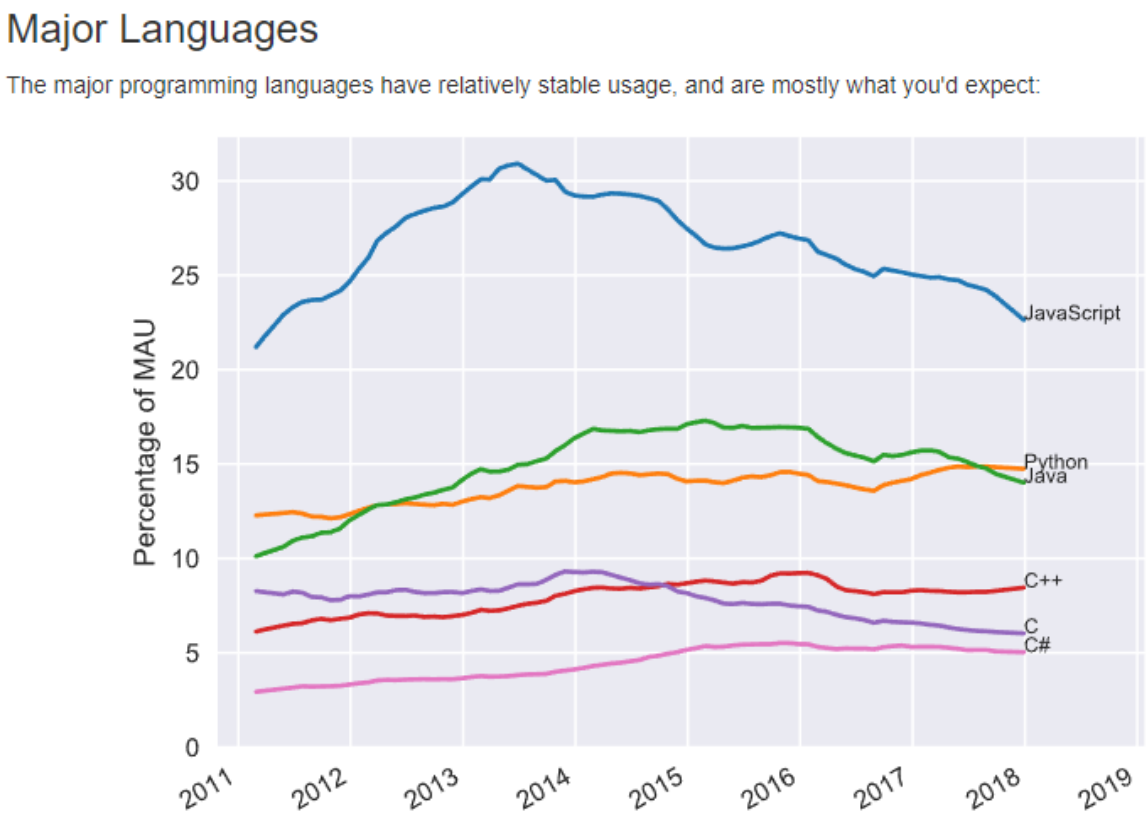
1. Complex numbers
2. Logic operations
3. Aggregation functions
   `min()` ; `max()` ; `sum()` ; `round()`

Complex numbers in different languages

Build-in type	Standard library	No Standard library
Fortran	C	Java
Python	C++	
Go	Perl	
Common Lisp	Ruby	
	OCaml	
	Haskell	

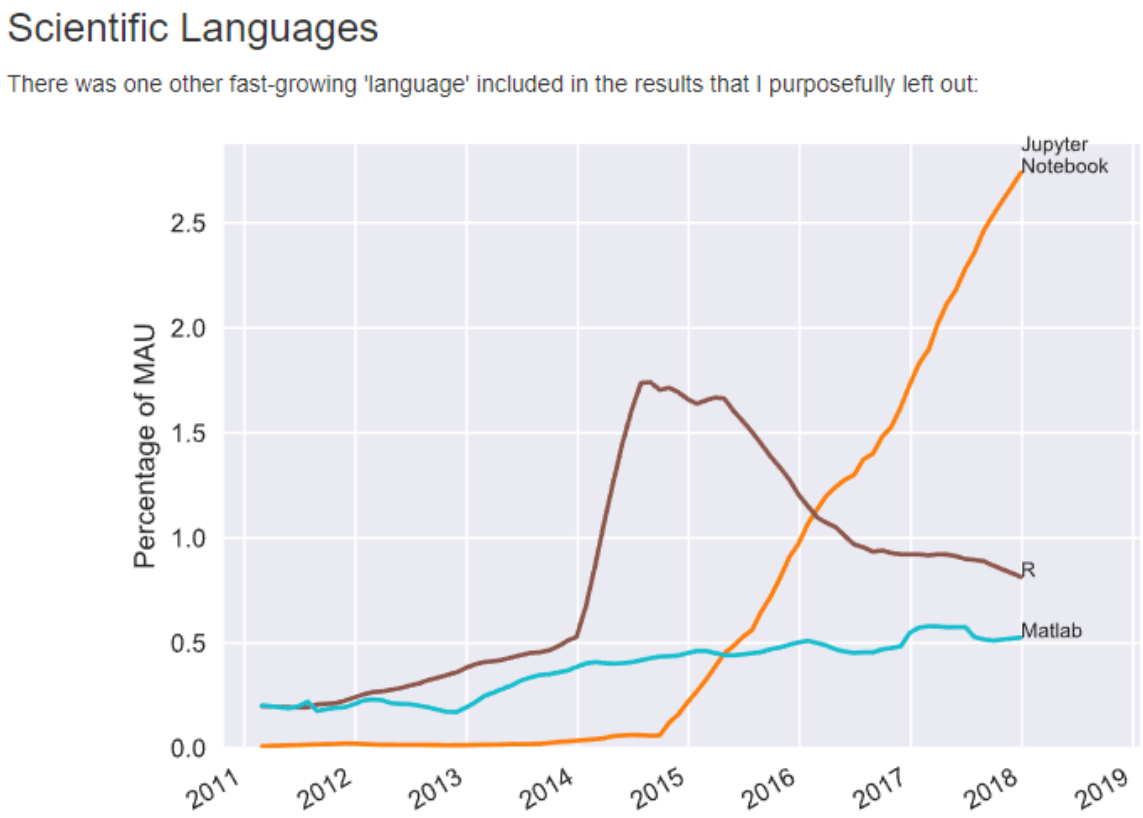
https://en.wikipedia.org/wiki/Complex_data_type

Python is super



https://www.benfrederickson.com

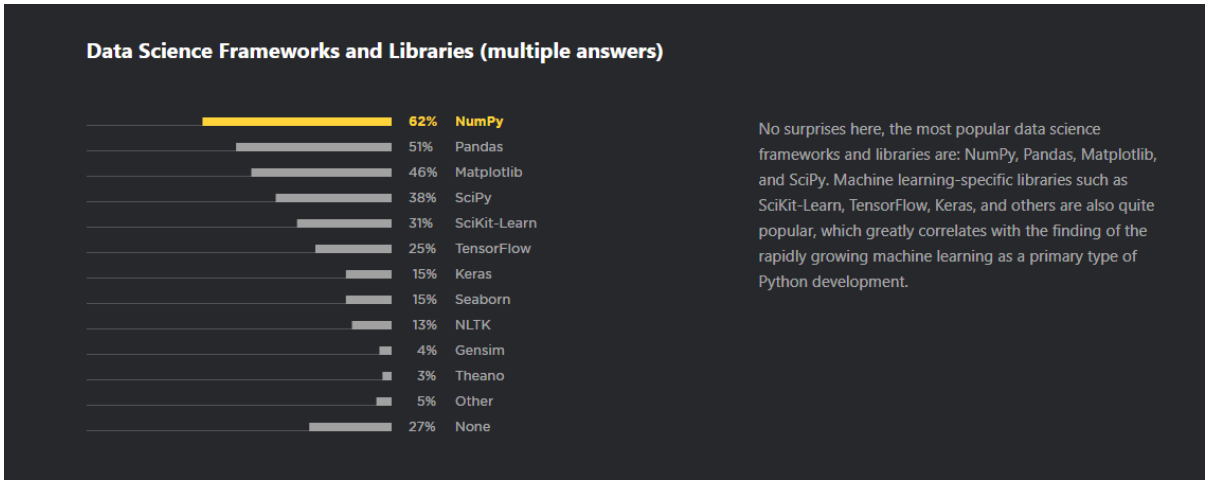
Numerical Python is superb



www.benfrederickson.com

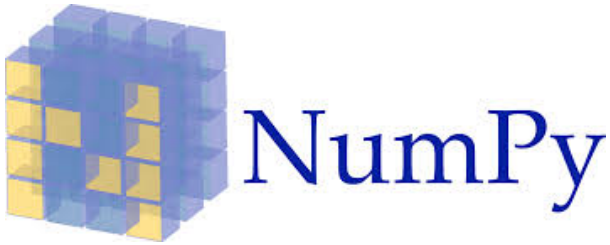
Table of Contents

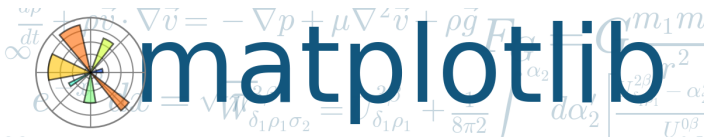
NumPy in Data Science



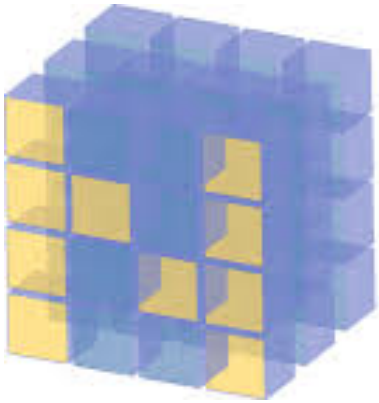
www.jetbrains.com/research/python-developers-survey-2018

Python vs Matlab





NumPy is Everything



NumPy

2447 packages depend on NumPy:

scipy, Keras, pandas, spacy, bokeh, dask, gym, netCDF4, numexpr, moviepy, textblob, nibabel, PyWavelets, autograd, tensorboardX, mxnet, folium, datasketch, rasterio, pymc3, ChatterBot, Bottleneck, thinc, OpenFisca-Core, holoviews, h5py, etc.

libraries.io/pypi/numpy/dependents

NumPy documentation

NumPy has a very rich official documentation, provided for both NumPy and SciPy

docs.scipy.org

Still, there is a huge number of books, courses, videos, and ect. Among numerous other authors I dare to sugeest Jacob VanderPlas. Here is a link to his page.

Numerical packages

NumPy

1. np.ndarray
2. np.random
3. basic Maths functions

SciPy

1. Special functions (scipy.special)
2. Integration (scipy.integrate)
3. Optimization (scipy.optimize)
4. Interpolation (scipy.interpolate)
5. Fourier Transforms (scipy.fftpack)
6. Signal Processing (scipy.signal)
7. Linear Algebra (scipy.linalg)
8. Sparse Eigenvalue Problems with ARPACK
9. Spatial data structures and algorithms (scipy.spatial)
10. Statistics (scipy.stats)

***numpy.ndarray**(shape, dtype, buffer, offset, strides, order)*

Parameters:

shape : *tuple of ints*

Shape of created array.

dtype : *data-type, optional*

Any object that can be interpreted as a numpy data type.

buffer : *object exposing buffer interface, optional*

Used to fill the array with data.

offset : *int, optional*

Offset of array data in buffer.

interpreted only if **buffer** is an object exposing the buffer interf

strides : *tuple of ints, optional*

Strides of data in memory.

interpreted only if **buffer** is an object exposing
the buffer interface

order : {*'C'*, *'F'*}, *optional*

Row-major (C-style) or column-major (Fortran-style) order.

interpreted only if **buffer** is an object exposing the buffer interf

NumPy supports a much greater variety of numerical types than Python does. They could be platform-dependent and independent. Here some former ones.

Numpy type	C type	Description
np.int8	int8_t	Byte(-128 to 127)
np.int16	int16_t	Integer(-32768 to 32767)
np.int32	int32_t	Integer(-2147483648 to 2147483647)
np.int64	int64_t	Integer (-9223372036854775808 to 9223372036854775807)
np.float32	float	
np.float64 / np.float_	double	precision of the builtin python float
np.complex64	float complex	Complex number, represented by two 32-bit floats (real and imaginary)

A universal function (or ufunc for short) is a function that operates on ndarrays in an element-by-element fashion, supporting array broadcasting, type casting, and several other standard features. That is, a ufunc is a "vectorized" wrapper for a function that takes a fixed number of specific inputs and produces a fixed number of specific outputs.

1. Arithmetic Operations: $+$ $-$ $*$ $@$ $/$ $//$ $\%$ $**$
2. Bitwise Operations: $\&$ $|$ $\backsim$ $\wedge$ $>>$ $<<$
3. Comparison Operations: $<$ $>$ $<=$ $=>$ $==$ $!=$
4. Mathematical Functions: np.sin, np.cos np.sqrt np.exp, np.log, np.log10, etc.
5. and many, many more

Thank you for your attantion :)

the presentation is here