Creating NumPy arrays

- **Load from file**: `np.load(`*filename*`)`,
  `np.loadtxt(`*filename*`)`
- **From Python object**: `np.array(`*list*`)`
- **Zeroes**: `np.zeros(`*shape*`,` *dtype*`)`,
  `np.zeros_like(`*shape*`)`
- A NumPy *shape* is a tuple of integers, one per dimension
- **Random**: `np.random.random(`*shape*`)`,
  `np.random.randint(`*min*`,` *max*`,` *shape*`)`,
  `np.random.standard_normal(`*shape*`)`

Slicing in NumPy

- **Basic Slicing**: `array[`*start*`:`*stop*`:`*step*`]` (omit any for shortcut)
- **Assignment**: `array[`*start*`:`*stop*`:`*step*`]` = *value* or *values*
- **Multi-Dimensional**: `array[`*start1*`:`*stop1*`:`*step1*`,`
  start2`:`*stop2*`:`*step2*`]`
- **Boolean Indexing**: `array[`*condition*`]` (e.g., `array[array > 0]`)
- **Fancy Indexing**: `array[`*array of indices*`]`

NumPy universal functions

- **Arithmetic**: `+`, `-`, `*`, `/`, `**`, `%` (modulus), `np.abs`, `np.sqrt`
- **Logical/bitwise**: `&` (and), `|` (or), `^` (xor), `~` (not)
- **Trigonometric**: `np.sin`, `np.cos`, `np.tan`, `np.arcsin`, `np.arccos`, `np.arctan`, `np.arctan2(y,x)`
- **Hyperbolic**: `np.sinh`, `np.cosh`, `np.tanh`, `np.arcsinh`, `np.arccosh`, `np.arctanh`
- **Exponential/logarithmic**: `np.exp`, `np.log`, `np.log10`
- **Miscellaneous**: `np.isinf`, `np.isnan`, `np.round`, `np.floor`, `np.ceil`
- ...

NumPy broadcasting

- NumPy matches dimension from the right
- NumPy adds dimensions on the left (with size one) when needed
- NumPy then *broadcasts* along size-one dimensions
- User can add dimensions explicitly with `np.newaxis`

$(3, 2) \star (3, 1) \rightarrow (3, 2)$

1	2	+	10	=	11	12
3	4		20		23	24
5	6		30		35	36

$(3, 2) \star (1, 2) \rightarrow (3, 2)$

$(3, 2) \star (2,) \rightarrow (3, 2) \star (1, 2) \rightarrow (3, 2)$

1	2	+	100	200	=	101	202
3	4					103	204
5	6					105	206

More NumPy features

- **Reshaping and combining**:
  `np.reshape`, `np.flatten`, `np.transpose`, `np.concatenate`, `np.vstack`, `np.hstack`
- **Aggregation and statistics**:
  `np.sum`, `np.prod`, `np.mean`, `np.median`, `np.std`, `np.var`, `np.min`, `np.max`, `np.argmin`, `np.argmax`, `np.polyfit`, `np.polyval`
- **Linear algebra**: `np.inv`, `np.det`, `np.eig`, `np.svd`, `np.solve`
- **Sorting/searching/counting**:
  `np.sort`, `np.argsort`, `np.where`, `np.unique`, `np.bincount`
- **Fourier transform**: `np.fft.fft`, `np.fft.ifft`

NumPy dtypes

dtype	description	dtype string	bytes
`np.int8`	8-bit integer	`'i1'`	1
`np.int16`	16-bit integer	`'i2'`	2
`np.int32`	32-bit integer	`'i4'` or `'i'`	4
`np.int64`	64-bit integer (Python int)	`'i8'`	8
`np.float32`	32-bit float	`'f4'` or `'f'`	4
`np.float64`	64-bit float (Python float)	`'f8'` or `'d'`	8
`np.bool_`	Boolean	`'?'`	1
(`np.str_`, *length*)	Unicode string	`'U`*length*`'`	4 x length (UTF-32)
(`np.bytes_`, *length*)	Byte string	`'S`*length*`'`	length
`np.datetime64`	np datetime	`'datetime64[`*unit*`]'` or `'M8[`*unit*`]'`	8
`np.timedelta64`	np datetime Δ	`'timedelta64[`*unit*`]'`	8