Python Ragged array library development

Oleksii Hrechykha

Mentors:
Ianna Osborne
Jim Pivarski

Awkward and ragged

- are time and memory-efficient
- take up less storage

- Awkward uses numpy and has a similar effect on performance
- but it allows for arbitrary data structures (i.e. supports numbers, dates, strings, record structures)
- ragged uses Awkward but is API-complacent

```
>>> pyobj = measure_memory(make_big_python_object)
memory: 2.687 GB

>>> arr = measure_memory(make_ragged_array)
memory: 0.877 GB
```

```
>>> result = measure_time(compute_on_python_object)
time: 4.180 sec

>>> result = measure_time(compute_on_ragged_array)
time: 0.082 sec
```

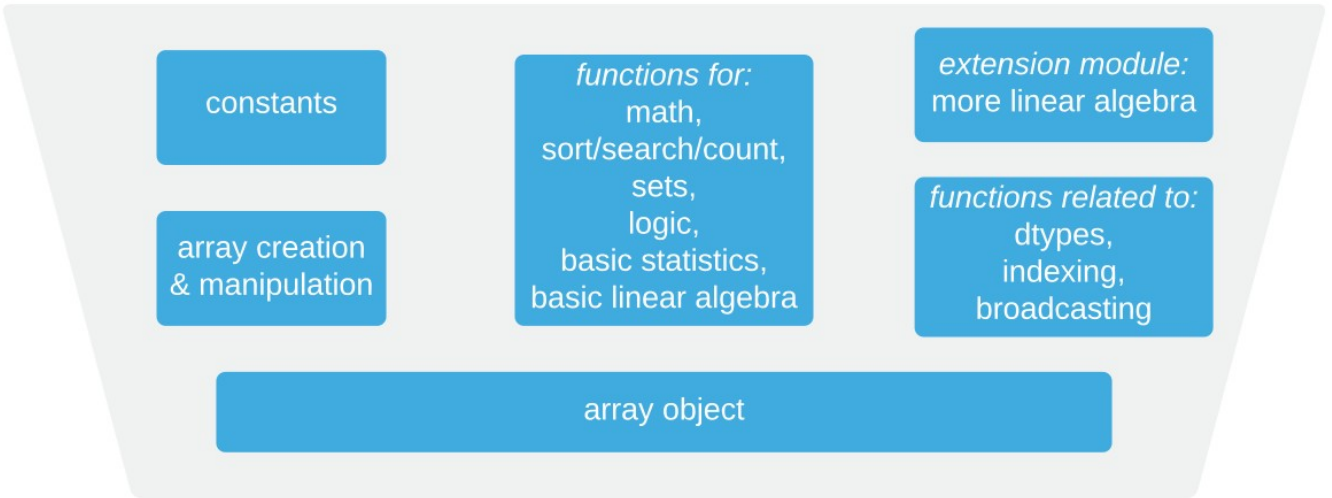
```
ragged.array([
    [[1.1, 2.2, 3.3], []],
    [[4.4]],
    [],
    [[5.5, 6.6, 7.7, 8.8], [9.9]]
])
```

```
>>> a.dtype
dtype('float64')
```

```
>>> a.shape
(4, None, None)
```

https://github.com/scikit-hep/ragged/blob/fff53544be6f9ded884e112d41ba476751b84085/README.md

Array API



Out of scope

Behavior and functions/objects that aren't common or cannot be fully specified are out of scope.

This doesn't mean libraries cannot implement them, only that they're not part of the standard.

- *I/O*
- *Polynomials*
- *Error handling*
- *Testing & building*
- *CFFI/ctypes*
- *Datetime dtypes*
- *String dtype*

https://data-apis.org/array-api/latest/purpose_and_scope.html

Example function

```python
def unique_values(x: array, /) -> array:
    """
    Returns the unique elements of an input array `x`.

    Args:
        x: Input array. If `x` has more than one dimension, the function
           flattens `x` and returns the unique elements of the flattened
           array.

    Returns:
        An array containing the set of unique elements in `x`. The returned
        array has the same data type as `x`.

    https://data-apis.org/array-api/latest/API_specification/generated/array_api.unique_values.html
    """

    x  # noqa: B018, pylint: disable=W0104
    raise NotImplementedError("TODO 131")  # noqa: EM101
```

- A function that returns the unique elements of an input array, the first occurring indices for each unique element in this array, the indices from the set of unique elements that reconstruct it, and the corresponding `counts` for each unique element ("TODO 128")
- A function that returns the unique elements of an input array and the corresponding counts for each unique element in this array ("TODO 129").
- A function that returns the unique elements of an input array and the indices from the set of unique elements that reconstruct it ("TODO 130")
- A function that returns the unique elements of an input array ("TODO 131").

Workflow so far:
1) Understand how the function works
2) Write an algorithm for simplified function
3) Write and run tests
4) Measure calculation time, compare it with NumPy

5) Write a better algorithm (and use Numba)
6) Implement the funcition
7) Write better tests

Thank you for your attention!