Arkouda Symbol Table and Symbol Table Entry

allows for passing results between operations

Symbol Table

- allows for passing results to future operations
- flat map between symbols(strings) and symbol table entries which are currently `pdarrays`
- add, delete, lookup, register, etc.
- `tab` holds `shared` entries... means they are reference counted
- `registry` holds registered symbols

Object refs

- discuss owned vs shared
  - `owned` is owned by the current scope and no one else, freed when goes out of scope
  - `shared` is shared with others and is reference counted, freed when reference count goes to 0
- `borrowed` is a reference which is from the owner or share-er

Chapel types

- type is only available at compile time (like `param`)
- not testable at runtime
- code is versioned/genertated for each type combination used in the chapel code
- if you want the type of some variable you can say `a.type`
- if you want to print a type use `a.type:string` to cast the type to a string
- types have a static component(compile-time) and sometimes a dynamic component(run-time)

Symbol Table Entry

- Generic Symbol Table Entry
- Specific Symbol Table Entries inherit from generic entry and are specialized by element type
- Chapel types are only available at compile time so we also needed the generic parent class to have a runtime testable element type, so we used the numpy dtype
- discuss `var aD: makeDistDom(size).type`
- two `init()` procedures, one creates a default initialized array and the other copies an array into a symbol table entry

look at code now