PythonDataStructures

Release 0.1

Christopher Neely

Jan 18, 2021

CONTENTS:

CHAPTER

ONE

PARALLEL ITERATION DOCUMENTATION

`data_structures.parallel_iter.`**`iter_threaded`**(*threads: int*, *ignore_types: Optional[Iterable[Optional[type]]] = None*, ***kwargs: Sequence*)

Parallelize function call using provided kwargs input dict. All args/kwargs not provided are not adjusted. Each kwarg passed is expected to be a subclass of Sequence, and all kwargs are expected to have the same input length.

Uses concurrent.futures and broadcasts calls across multiple threads

Parameters

- **`threads`** – Number of threads to launch to complete task list
- **`ignore_types`** – Iterable of return types/Exception types to handle in parallelized call
- **`kwargs`** – Keyword arguments to override in function

Raises AttributeError for improperly formatted input data

Returns Generator over results from each parallelized function call (in order)

`data_structures.parallel_iter.`**`iter_async`**(***kwargs: Sequence*)

Parallelize function call using provided kwargs input dict. All non-kwargs are not adjusted. Expected input is dict mapping to list of inputs to try.

Must decorate an async def function for valid functionality

Uses asyncio and maintains call running over single thread

Parameters **`kwargs`** – Keyword arguments to override in function

Raises AttributeError for improperly formatted input data

Returns Decorated function

CHAPTER

TWO

TYPECHECKER DOCUMENTATION

`data_structures.type_checking.TypeChecker.__call__`(*func: Callable*)

Check if types of args/kwargs passed to function/method are valid for provided type signatures

Parameters **`func`** – Called function/method

Raises TypeError for improper arg/kwarg type combinations

Returns Decorated function/method. Raises TypeError if improper type/arg combination is found

CHAPTER

THREE

MUTABLE STRING DOCUMENTATION

Module holds class functionality for mutable strings

class `data_structures.mutable_string.`**`Str`**(*string: Union[str, data_structures.mutable_string.Str], const: bool = False*)

Mutable string class, pass-by-reference internal data

Pass by "const reference" ensures that inner data is not mutated, but does no further optimization than the standard Python pass-by-reference

Constructor inherently is deep copy-constructor

`clear`()

Clears contents of stored buffer

property const

Check if Str is const or non-const reference

Returns bool of const status

`format`(**args, **kwargs*) → *data_structures.mutable_string.Str*

Mimic str class format function

Parameters

- **args** – kwargs to format
- **kwargs** – kwargs to format

Returns Formatted Str object

`set_const`()

Sets const status of owned object to True

`split`(**args, **kwargs*) → List[*data_structures.mutable_string.Str*]

Split contents into python's str type

Parameters

- **args** – str.split() args
- **kwargs** – str.split() kwargs

Returns List of split strings

`data_structures.mutable_string.`**`handle_const`**(*func: Callable*)

Decorator checks if Str object is non-const reference

Parameters **func** – Str method to call

Raises TypeError for attempt to modify const value

Returns Wrapped method that has first checked if Str is mutable

This repository contains a few simple data structures that are (semi) useful in various applications. This code base is entirely in development and comes as-is.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

INDEX