Simple Linked List

In Practice Mode

Introduction

Write a simple linked list implementation that uses Elements and a List.

The linked list is a fundamental data structure in computer science, often used in the implementation of other data structures. They're pervasive in functional programming languages, such as Clojure, Erlang, or Haskell, but far less common in imperative languages such as Ruby or Python.

The simplest kind of linked list is a singly linked list. Each element in the list contains data and a "next" field pointing to the next element in the list of elements.

This variant of linked lists is often used to represent sequences or push-down stacks (also called a LIFO stack; Last In, First Out).

As a first take, lets create a singly linked list to contain the range (1..10), and provide functions to reverse a linked list and convert to and from arrays.

When implementing this in a language with built-in linked lists, implement your own abstract data type.

Hints

To support `list()`, see implementing an iterator for a class.

Additionally, note that Python2's `next()` has been replaced by `__next__()` in Python3. For dual compatibility, `next()` can be implemented as:

```
def next(self):
    return self.__next__()
```

Exception messages

Sometimes it is necessary to raise an exception. When you do this, you should include a meaningful error message to indicate what the source of the error is. This makes your code more readable and helps significantly with debugging. Not every exercise will require you to raise an exception, but for those that do, the tests will only pass if you include a message.

To raise a message with an exception, just write it as an argument to the exception type. For example, instead of `raise Exception`, you should write:

```
raise Exception("Meaningful message indicating the source of the error")
```

Running the tests

To run the tests, run `pytest simple_linked_list_test.py`

Alternatively, you can tell Python to run the pytest module: `python -m pytest simple_linked_list_test.py`

Common `pytest` options

- `-v` : enable verbose output
- `-x` : stop running tests on first failure
- `--ff` : run failures from previous test before running other test cases

For other options, see `python -m pytest -h`

Submitting Exercises

Note that, when trying to submit an exercise, make sure the solution is in the `$EXERCISM_WORKSPACE/python/simple-linked-list` directory.

You can find your Exercism workspace by running `exercism debug` and looking for the line that starts with `Workspace`.

For more detailed information about running tests, code style and linting, please see Running the Tests.

Source

Inspired by 'Data Structures and Algorithms with Object-Oriented Design Patterns in Ruby', singly linked-lists. http://www.brpreiss.com/books/opus8/html/page96.html#SECTION00430000000000000000

Submitting Incomplete Solutions

It's possible to submit an incomplete solution so you can see how others have completed the exercise.

Get started

We've created a step-by-step set of instructions to help you quickly install Exercism and get started, so you can start learning.

Begin walk-through

1. Download

```
exercism download --exercise=simple-lin
```
copy

2. Solve

Use your code editor to solve the problem locally.

3. Submit

```
exercism submit /path/to/file [/path/to
```
copy

Still stuck?

These guides should help you.

- Installing Python
- Running the Tests
- Learning Python
- Useful Python Resources
- Remind me how to set up the CLI