Effectively Using Arrays And Lists: Takeaways

Syntax

- Converting an integer to binary:

```
bin(10)
```

- Converting a number as a string to a decimal integer:

```
int("1010", 2)
```

- Retrieving the hexadecimal memory address of any variable:

```
hex(id(1))
```

- Implementing an array as a class:

```
import numpy as np

class Array():

     def __init__(self, size):

         self.array = np.zeros(size, dtype=np.float64)

         self.size = size
```

- Accessing and setting items in a class:

```
def __getitem__(self, key):

    return self.array[key])

def __setitem__(self, key):

    return.self(array[key])
```

- Inserting items to an list:

```
list = [1,2,3]

list.insert(1,5) # list is now [1,5,2,3]
```

Concepts

- Binary is a convenient way to store data since you only need to store two "positions".
- Python lists are implemented as C arrays but lists don't have a fixed size, and they can store elements of any type.
- A pointer is a special kind of variable in C that points to the value of another variable in memory. Pointers allow us to refer to the same value in multiple places without having to copy the value.
- The NumPy array type is based on a C array and behaves very similarly, so it's a better choice for implementing an array class than a Python list.
- Linked lists allow you to flexibly add as many elements as desired. Linked lists achieve this by storing links between items.
- Linked lists don't allow for directly indexing an item like in an array. Instead, we need to scan through the list to find the item we want.
- Linked lists have the following characteristics:
    - You don't have to specify how many nodes you want upfront — you can store as many values as you want.
    - Data isn't restricted to a single type — you can store any data you want in any node.
    - Finding an element in a linked list has time complexity $O(n)$ time since we need to iterate through all of the elements to find the one we want.
    - Insertions and deletions are fast since we don't need to copy anything — we just need to find the insertion or deletion point.
- Linked lists are better if you're storing values, and you don't know how many you want to store. Linked lists are also easier to combine and shuffle, which makes them very useful when gathering data.
- Arrays are better when you need to access data quickly, but won't be changing it much. Arrays are usually much better for computation, such as when you're analyzing data.

Resources

- Python List Implementation
- NumPy Arrays

- Linked Lists
- Arrays