Lists and Arrays in Python

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 3

Lists and arrays in Python

- Sequences can be stored as lists or arrays
- Lists are flexible but accessing an element is $O(n)$
- Arrays support random access but are difficult to expand, contract
- Algorithm analysis needs to take into account the underlying implementation
- How does it work in Python?
  - Is the built-in list type in Python really a "linked" list?
  - Numpy library provides arrays — are these faster than lists?

Lists in Python

- Python lists are not implemented as flexible linked lists

Lists in Python

- Python lists are not implemented as flexible linked lists
- Underlying interpretation maps the list to an array
  - Assign a fixed block when you create a list
  - Double the size if the list overflows the array

Lists in Python

- Python lists are not implemented as flexible linked lists
- Underlying interpretation maps the list to an array
  - Assign a fixed block when you create a list
  - Double the size if the list overflows the array
- Keep track of the last position of the list in the array
  - `l.append()` and `l.pop()` are constant time, amortised — $O(1)$
  - Insertion/deletion require time $O(n)$

Lists in Python

- Python lists are not implemented as flexible linked lists
- Underlying interpretation maps the list to an array
  - Assign a fixed block when you create a list
  - Double the size if the list overflows the array
- Keep track of the last position of the list in the array
  - `l.append()` and `l.pop()` are constant time, amortised — $O(1)$
  - Insertion/deletion require time $O(n)$
- Effectively, Python lists behave more like arrays than lists

Arrays vs lists in Python

- Arrays are useful for representing matrices
- In list notation, these are nested lists $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ `[[0,1], [1,0]]`

Arrays vs lists in Python

- Arrays are useful for representing matrices
- In list notation, these are nested lists $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ `[[0,1], [1,0]]`
- Need to be careful when initializing a multidimensional list

```
zerolist = [0,0,0]
zeromatrix = [zerolist,zerolist,zerolist]
```

Arrays vs lists in Python

- Arrays are useful for representing matrices
- In list notation, these are nested lists $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ `[[0,1], [1,0]]`
- Need to be careful when initializing a multidimensional list

```
zerolist = [0,0,0]
zeromatrix = [zerolist,zerolist,zerolist]

zeromatrix[1][1] = 1
print(zeromatrix)
```

Arrays vs lists in Python

- Arrays are useful for representing matrices
- In list notation, these are nested lists $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ `[[0,1], [1,0]]`
- Need to be careful when initializing a multidimensional list

```
zerolist = [0,0,0]
zeromatrix = [zerolist,zerolist,zerolist]

zeromatrix[1][1] = 1
print(zeromatrix)

[[0, 1, 0], [0, 1, 0], [0, 1, 0]]
```

- Mutability aliases different values

Arrays vs lists in Python

- Arrays are useful for representing matrices
- In list notation, these are nested lists $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ `[[0,1], [1,0]]`
- Need to be careful when initializing a multidimensional list

```
zerolist = [0,0,0]
zeromatrix = [zerolist,zerolist,zerolist]

zeromatrix[1][1] = 1
print(zeromatrix)

[[0, 1, 0], [0, 1, 0], [0, 1, 0]]
```

- Mutability *aliases* different values
- Instead, use list comprehension

```
zeromatrix = [ [ 0 for i in range(3) ]  for j in range(3) ]
```

Numpy arrays

- The Numpy library provides arrays as a basic type

```
import numpy as np
zeromatrix = np.zeros(shape=(3,3))
```

Numpy arrays

- The Numpy library provides arrays as a basic type

```
import numpy as np
zeromatrix = np.zeros(shape=(3,3))
```

- Can create an array from any sequence type

```
identitymatrix = np.array([[1,0],[0,1]])
```

Numpy arrays

- The Numpy library provides arrays as a basic type

```
import numpy as np
zeromatrix = np.zeros(shape=(3,3))
```

- Can create an array from any sequence type

```
identitymatrix = np.array([[1,0],[0,1]])
```

- `arange` is the equivalent of `range` for lists

```
row2 = np.arange(5)
```

Numpy arrays

- The Numpy library provides arrays as a basic type

```
import numpy as np
zeromatrix = np.zeros(shape=(3,3))
```

- Can create an array from any sequence type

```
identitymatrix = np.array([[1,0],[0,1]])
```

- `arange` is the equivalent of `range` for lists

```
row2 = np.arange(5)
```

- Can operate on a matrix as a whole
  - `C = 3*A + B`
  - `C = np.matmul(A,B)`
  - Very useful for data science

Summary

- Python lists are not implemented as flexible linked structures
- Instead, allocate an array, and double space as needed
- Append is cheap, insert is expensive
- Arrays can be represented as multidimensional lists, but need to be careful about mutability, aliasing
- Numpy arrays are easier to use