Coming up:

Due Mon, Sept 12: (end of the day)
Read (or finish reading ) Section 2.(2, 4, & 5)
Take data structures quiz

Also:
Do “basic data structures” practice problems (suggested by Wed, Sept 14)
Do “Implementing ADTs” project (suggested by Fri, Sept 16)

Due Fri, Sept 16: (class time)
Read Section 3.1
Do Exercises 2.(22–24)
Take sorting quiz
This week and next week (Chapters 2 and 3):

- Abstract data types (Wednesday)
- Data Structures (today and Monday)
- Programming practices (Monday)
- Linear time sorting (next week Wednesday and Friday)

Today:

- Ex 1.11
- Recent quiz problems
- ADT review
- Data structure categories
- List vs array
- Abstractions
- Adapter pattern
def is_palindrome(str):
    palindromic = True
    n = len(str)
i = 0
    while palindromic and i < n // 2:
        palindromic = str[i] == str[n-i-1]
i += 1
    return palindromic

Invariant (Loop of is_palindrome)

1. $\forall j \in [0, i-1), str[j] = str[n-j-1]$
2. $palindromic \text{ iff } (i = 0 \text{ or } str[i-1] = str[n-i])$
3. $i$ is the number of iterations completed
<table>
<thead>
<tr>
<th></th>
<th>best case</th>
<th>worst case</th>
<th>expected case</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>binary search</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>bounded linear search</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>selection sort</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>merge sort</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>quick sort</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The “canonical ADTs”:

- **List.** Linear collection with sequential and random access.
- **Stack.** Linear collection with LIFO access.
- **Queue.** Linear collection with FIFO access.
- **Set.** Unordered collection with binary membership.
- **Bag.** Unordered collection with enumerated membership.
- **Map.** Unordered collection of associations between keys and values.
The four basic ways to implement an ADT:

- Use an array
- Use a linked structure
- Use an “advanced” data structure, varying and/or hybridizing linked structures and arrays
- Adapt an existing implementation of another ADT.
Abstract data type

Simple data structure

Advanced data structure

Abstraction

Simple data structure

Array queue data structure

Ring buffer abstraction

Queue ADT

Array data structure
Client
Adapter
operationA()
<<interface>>
Target
original
operationA()
original.operationB()
Original
operationB()
original.operationB()
Coming up:

Due Mon, Sept 12: (end of the day)
Read (or finish reading ) Section 2.(2, 4, & 5)
Take data structures quiz

Also:
Do “basic data structures” practice problems (suggested by Wed, Sept 14)
Do “Implementing ADTs” project (suggested by Fri, Sept 16)

Due Fri, Sept 16: (class time)
Read Section 3.1
Do Exercises 2.(22–24)
Take sorting quiz