Coming up:

Due Fri, Feb 28: (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 Mon, Jan 31)
Do “implementing ADTs” project (suggested by Wed, Feb 2)

Due Wed, Feb 2: (class time)
Read Section 3.1
Do Exercises 2.(21–23)
Take sorting quiz
This week (Chapter 2):
- Abstract data types (Monday)
- Data Structures (today and Friday)
- Programming practices (Friday)
- Start Chapter 3 Case studies (Next week)

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. $palindromic$ iff $(i = 0$ or $str[i - 1] = str[n - i - 1])$
3. $i$ is the number of iterations completed
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>best case</th>
<th>worst case</th>
<th>expected case</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary search</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bounded linear search</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>selection sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>merge sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quick sort</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.
List
Stack
LIFO access
Queue
Random access
FIFO access
Map
Lookup by index
Lookup by key
Whole-number value
Any value
Set
Bag
Items
Associations
Binary membership
Enumerated membership
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

Queue ADT

Array queue data structure
Ring buffer abstraction
Array data structure
Queue
<<interface>>
enqueue(E)
front()
remove()
isEmpty()

ListQueue
– internal:List
enqueue(E)
front()
remove()
isEmpty()

enqueue(E)
front()
remove()
isEmpty()
internal.add(item);
<<interface>>
push(E)
top()
pop()
isEmpty()
Stack
<<interface>>
push(E)
top()
pop()
isEmpty()

List
<<interface>>
add(E)
set(int,E)
get(int)
remove(int)
insert(int,E)
size()

ListQueue
– internal:List
enqueue(E)
front()
remove()
isEmpty()

ListStack
– internal:List
push(E)
top()
pop()
isEmpty()
Coming up:

Due Fri, Feb 28: (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 Mon, Jan 31)
Do “implementing ADTs” project (suggested by Wed, Feb 2)

Due Wed, Feb 2: (class time)
Read Section 3.1
Do Exercises 2.(21–23)
Take sorting quiz