Data Examples
Announcements
Examples: Objects
Land Owners

Instance attributes are found before class attributes; class attributes are inherited
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

```python
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
```
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

```python
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
```
Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()  >>> Worker().work()
john = Bourgeoisie()  >>> john.work()
jack.greeting = 'Maam'

jack => 'Maam', I work
john => 'Peon', I gather wealth
Land Owners

Instance attributes are found before class attributes; class attributes are inherited.

```python
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
```
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

```python
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
```

```python
>>> Worker().work()  
<class Worker>
    greeting: 'Sir'

>>> jack
<class Bourgeoisie>
    greeting: 'Peon'

>>> jack.work()  

>>> john.work()  

>>> john.elf.work(john)
```
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
greeting = 'Sir'
def __init__(self):
    self.elf = Worker
def work(self):
    return self.greeting + ', I work'
def __repr__(self):
    return Bourgeoisie.greeting

class Bourgeoisie(Worker):
greeting = 'Peon'
def work(self):
    print(Worker.work(self))
    return 'I gather wealth'

jack = Worker()  
john = Bourgeoisie()  
jack.greeting = 'Maam'

>>> Worker().work()  
<class Worker>  
greeting: 'Sir'

>>> jack  
<class Bourgeoisie>  
greeting: 'Peon'

>>> jack.work()  
jack <Worker>  
elf:

>>> john.work()  

>>> john.elf.work(john)
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
**Land Owners**

Instance attributes are found before class attributes; class attributes are inherited

```python
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()  # <class Worker>
    greeting: 'Sir'

>>> jack
    # <class Bourgeoisie>
    greeting: 'Peon'

>>> jack.work()
    jack <Worker>
    elf:
    greeting: 'Maam'

>>> john.work()
    # <class Bourgeoisie>
    greeting: 'Maam'

>>> john.elf.work(john)
    john <Bourgeoisie>
    elf:
```

---

4
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()
'Sir, I work'

>>> jack
<class Worker>
greeting: 'Sir'

>>> jack.work()
<class Bourgeoisie>
greeting: 'Peon'

>>> john.work()

>>> john.elf.work(john)

jack <Worker>
elf: 
greeting: 'Maam'

john <Bourgeoisie>
elf: 
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return 'Bourgeoisie'+self.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()
'Sir, I work'

>>> jack
<class Worker>
    greeting: 'Sir'

>>> jack.work()

>>> john.work()

>>> john.elf.work(john)

jack <Worker>
    elf:
    greeting: 'Maam'

john <Bourgeoisie>
    elf:
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

```python
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return 'I gather wealth'

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
```

```
>>> Worker().work()
'Sir, I work'

>>> jack
Peon

>>> jack.work()

>>> john.work()

>>> john.elf.work(john)

Jack <Worker>
    greeting: 'Sir'

Jack <Worker>
    elf:
    greeting: 'Maam'

John <Bourgeoisie>
    elf: 
    greeting: 'Maam'
```

```
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()
'Sir, I work'

>>> jack
Peon

>>> jack.work()

>>> john.work()

>>> john.elf.work(john)
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return 'Bourgeoisie.greeting'

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()
'Sir, I work'

>>> jack
Peon

>>> jack.work()
'Maam, I work'

>>> john.work()

>>> john.elf.work(john)

<class Worker>
greeting: 'Sir'

<class Bourgeoisie>
greeting: 'Peon'

jack <Worker>
elf: 
greeting: 'Maam'

john <Bourgeoisie>
elf: 

Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return self.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()
'Sir, I work'

>>> jack
Peon

>>> jack.work()
'Maam, I work'

>>> john.work()

>>> john.elf.work(john)

call Worker()<class Worker>
greeting: 'Sir'

call Bourgeoisie<class Bourgeoisie>
greeting: 'Peon'

call Worker()jack <Worker>
greeting: 'Peon'
elf: <class Worker>
geeling: 'Sir'

call Worker()john <Bourgeoisie>
greeting: 'Maam'
elf: <class Worker>
geeling: 'Sir'

call Bourgeoisie()john <Bourgeoisie>
greeting: 'Maam'
elf: <class Worker>
geeling: 'Sir'
Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return 'Bourgeoisie.' + self.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()
'Sir, I work'

>>> jack
'Peon'

>>> jack.work()
'Maam, I work'

>>> john.work()
'Peon, I work
'I gather wealth'

>>> john.elf.work(john)

<class Worker>
greeting: 'Sir'

<class Bourgeoisie>
greeting: 'Peon'

jack <Worker>
elf: 
greeting: 'Maam'

john <Bourgeoisie>
elf: 

Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()
'Sir, I work'

Jack
Peon

>>> jack
<class Worker>
greeting: 'Sir'

jack <Worker>
greeting: 'Peon'

>>> jack.work()
'Maam, I work'

>>> john.work()
'Peon, I work
'I gather wealth'

>>> john.elf.work(john)
<class Bourgeoisie>
greeting: 'Maam'

john <Bourgeoisie>
elf: ___

greeting: 'Maam'

jack <Worker>
elf: ___

greeting: 'Sir'

jack <Worker>

Land Owners

Instance attributes are found before class attributes; class attributes are inherited

class Worker:
    greeting = 'Sir'
def __init__(self):
    self.elf = Worker
def work(self):
    return self.greeting + ', I work'
def __repr__(self):
    return Bourgeoisie.greeting
class Bourgeoisie(Worker):
    greeting = 'Peon'
def work(self):
    print(Worker.work(self))
    return 'I gather wealth'

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'

>>> Worker().work()
'Sir, I work'

>>> jacks
Peon

>>> jack.work()
'Maam, I work'

>>> john.work()
'Peon, I work'

>>> john.elf.work(john)
'Peon, I work'
Examples: Iterables & Iterators
Using Built-In Functions & Comprehensions
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list `s` that have the smallest absolute value?
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list \( s \) that have the smallest absolute value?

\([-4, -3, -2, 3, 2, 4]\)
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?

$[-4, -3, -2, 3, 2, 4]$

\[0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5\]
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?

$[-4, -3, -2, 3, 2, 4]$  $\Rightarrow$  $[2, 4]$
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list \( s \) that have the smallest absolute value?

\[
[-4, -3, -2, 3, 2, 4]
\]

\[
0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5
\]

\[ [2, 4] \quad [1, 2, 3, 4, 5] \]
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?

$[-4, -3, -2, 3, 2, 4]$  $[2, 4]$  $[1, 2, 3, 4, 5]$  $[0]$
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?

\[
[-4, -3, -2, 3, 2, 4] \quad \Rightarrow \quad [2, 4] \quad [1, 2, 3, 4, 5] \quad \Rightarrow \quad [0]
\]

What's the largest sum of two adjacent elements in a list $s$? (Assume $\text{len}(s) > 1$)
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?

\[-4, -3, -2, 3, 2, 4\]

$0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$ $\quad [2, 4]$ $\quad [1, 2, 3, 4, 5]$ $\quad [0]$

What's the largest sum of two adjacent elements in a list $s$? (Assume len($s$) > 1)

\[-4, -3, -2, 3, 2, 4\]
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list \( s \) that have the smallest absolute value?

\[
[-4, -3, -2, 3, 2, 4] \quad [0, 1, 2, 3, 4, 5] \quad [2, 4] \quad [1, 2, 3, 4, 5] \quad [0]
\]

What's the largest sum of two adjacent elements in a list \( s \)? (Assume \( \text{len}(s) > 1 \))

\[
[-4, -3, -2, 3, 2, 4] \quad 6
\]
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list \( s \) that have the smallest absolute value?

\[
[-4, -3, -2, 3, 2, 4]
\]

\[
0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5
\]

\[
[2, 4] \quad [1, 2, 3, 4, 5] \quad [0]
\]

What's the largest sum of two adjacent elements in a list \( s \)? (Assume \( \text{len}(s) > 1 \))

\[
[-4, -3, -2, 3, 2, 4]
\]

\[
0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5
\]

\[
6 \quad [-4, 3, -2, -3, 2, -4]
\]
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list s that have the smallest absolute value?

\[-4, -3, -2, 3, 2, 4\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\[
[2, 4] \\
[1, 2, 3, 4, 5] \\
[0]
\]

What's the largest sum of two adjacent elements in a list s? (Assume len(s) > 1)

\[-4, -3, -2, 3, 2, 4\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\[
[6] \\
[1]
\]

\[-4, 3, -2, -3, 2, -4\]
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list s that have the smallest absolute value?

\[
[-4, -3, -2, 3, 2, 4] \\
0 1 2 3 4 5 \quad \Rightarrow \quad [2, 4] \quad \quad [1, 2, 3, 4, 5] \quad \Rightarrow \quad [0]
\]

What's the largest sum of two adjacent elements in a list s? (Assume len(s) > 1)

\[
[-4, -3, -2, 3, 2, 4] \quad \Rightarrow \quad 6 \quad \quad [-4, 3, -2, -3, 2, -4] \quad \Rightarrow \quad 1
\]

Create a dictionary mapping each digit d to the lists of elements in s that end with d.
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?

$[-4, -3, -2, 3, 2, 4]$  $[2, 4]$  $[1, 2, 3, 4, 5]$  $[0]$

What's the largest sum of two adjacent elements in a list $s$? (Assume $\text{len}(s) > 1$)

$[-4, -3, -2, 3, 2, 4]$  $6$  $[-4, 3, -2, -3, 2, -4]$  $1$

Create a dictionary mapping each digit $d$ to the lists of elements in $s$ that end with $d$.

$[5, 8, 13, 21, 34, 55, 89]$
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list \( s \) that have the smallest absolute value?

\[
[-4, -3, -2, 3, 2, 4] \\
0 1 2 3 4 5 \\
\Rightarrow [2, 4] \\
[1, 2, 3, 4, 5] \\
\Rightarrow [0]
\]

What's the largest sum of two adjacent elements in a list \( s \)? (Assume \( \text{len}(s) > 1 \))

\[
[-4, -3, -2, 3, 2, 4] \\
\Rightarrow 6 \\
[-4, 3, -2, -3, 2, -4] \\
\Rightarrow 1
\]

Create a dictionary mapping each digit \( d \) to the lists of elements in \( s \) that end with \( d \).

\[
[5, 8, 13, 21, 34, 55, 89] \\
\Rightarrow \{1: [21], 3: [13], 4: [34], 5: [5, 55], 8: [8], 9: [89]\}
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list s that have the smallest absolute value?

\[-4, -3, -2, 3, 2, 4\]  
\[0, 1, 2, 3, 4, 5\]  
\[2, 4\]  
\[1, 2, 3, 4, 5\]  
\[0\]

What's the largest sum of two adjacent elements in a list s? (Assume \(\text{len}(s) > 1\))

\[-4, -3, -2, 3, 2, 4\]  
6  
\[-4, 3, -2, -3, 2, -4\]  
1

Create a dictionary mapping each digit d to the lists of elements in s that end with d.

\[5, 8, 13, 21, 34, 55, 89\]  
\{1: [21], 3: [13], 4: [34], 5: [5, 55], 8: [8], 9: [89]\}

Does every element equal some other element in s?
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list \( s \) that have the smallest absolute value?

\[
[-4, -3, -2, 3, 2, 4] \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad \Rightarrow \quad [2, 4] \quad [1, 2, 3, 4, 5] \quad \Rightarrow \quad [0]
\]

What's the largest sum of two adjacent elements in a list \( s \)? (Assume \( \text{len}(s) > 1 \))

\[
[-4, -3, -2, 3, 2, 4] \quad \Rightarrow \quad 6 \quad [-4, 3, -2, -3, 2, -4] \quad \Rightarrow \quad 1
\]

Create a dictionary mapping each digit \( d \) to the lists of elements in \( s \) that end with \( d \).

\[
[5, 8, 13, 21, 34, 55, 89] \quad \Rightarrow \quad \{1: [21], 3: [13], 4: [34], 5: [5, 55], 8: [8], 9: [89]\}
\]

Does every element equal some other element in \( s \)?

\[
[-4, -3, -2, 3, 2, 4] \quad \Rightarrow \quad \text{False}
\]
Using Built-In Functions & Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?

$$[-4, -3, -2, 3, 2, 4] \rightarrow [2, 4] \quad [1, 2, 3, 4, 5] \rightarrow [0]$$

What's the largest sum of two adjacent elements in a list $s$? (Assume len($s$) > 1)

$$[-4, -3, -2, 3, 2, 4] \rightarrow 6 \quad [-4, 3, -2, -3, 2, -4] \rightarrow 1$$

Create a dictionary mapping each digit $d$ to the lists of elements in $s$ that end with $d$.

$$[5, 8, 13, 21, 34, 55, 89] \rightarrow \{1: [21], 3: [13], 4: [34], 5: [5, 55], 8: [8], 9: [89]\}$$

Does every element equal some other element in $s$?

$$[-4, -3, -2, 3, 2, 4] \rightarrow \text{False} \quad [4, 3, 2, 3, 2, 4] \rightarrow \text{True}$$
Examples: Linked Lists
Linked List Exercises
Linked List Exercises

Is a linked list ordered from least to greatest?
Is a linked list ordered from least to greatest?

1 → 3 → 4
Is a linked list s ordered from least to greatest?
Linked List Exercises

Is a linked list $s$ ordered from least to greatest?

![Linked List Diagram 1]

![Linked List Diagram 2]

Is a linked list $s$ ordered from least to greatest by absolute value (or a key function)?
Linked List Exercises

Is a linked list s ordered from least to greatest?

1 → 3 → 4
1 → 4 → 3

Is a linked list s ordered from least to greatest by absolute value (or a key function)?

1 → -3 → 4
Linked List Exercises

Is a linked list $s$ ordered from least to greatest?

\begin{align*}
1 & \rightarrow 3 & \rightarrow 4 \\
1 & \rightarrow 4 & \rightarrow 3
\end{align*}

Is a linked list $s$ ordered from least to greatest by absolute value (or a key function)?

\begin{align*}
1 & \rightarrow -3 & \rightarrow 4 \\
1 & \rightarrow 4 & \rightarrow -3
\end{align*}
Linked List Exercises

Is a linked list $s$ ordered from least to greatest?

Is a linked list $s$ ordered from least to greatest by absolute value (or a key function)?

Create a sorted Link containing all the elements of both sorted Links $s$ & $t$. 
**Linked List Exercises**

Is a linked list $s$ ordered from least to greatest?

![Linked List Diagram 1](image1)

Is a linked list $s$ ordered from least to greatest by absolute value (or a key function)?

![Linked List Diagram 2](image2)

Create a sorted Link containing all the elements of both sorted Links $s$ & $t$.

![Linked List Diagram 3](image3)
## Linked List Exercises

Is a linked list $s$ ordered from least to greatest?

![Linked List Diagram](image)

Is a linked list $s$ ordered from least to greatest by absolute value (or a key function)?

![Linked List Diagram](image)

Create a sorted Link containing all the elements of both sorted Links $s$ & $t$.

![Linked List Diagram](image)
Linked List Exercises

Is a linked list $s$ ordered from least to greatest?

```
1 --> 3 --> 4
1 --> 4 --> 3
```

Is a linked list $s$ ordered from least to greatest by absolute value (or a key function)?

```
1 --> -3 --> 4
1 --> 4 --> -3
```

Create a sorted Link containing all the elements of both sorted Links $s$ & $t$.

```
1 --> 5
1 --> 4
1 --> 1 --> 4 --> 5
```

Do the same thing, but never call Link.
Linked List Exercises

Is a linked list $s$ ordered from least to greatest?

![Diagram of linked list with elements 1, 3, 4, 1, 4, 3]

Is a linked list $s$ ordered from least to greatest by absolute value (or a key function)?

![Diagram of linked list with elements 1, -3, 4, 1, 4, -3]

Create a sorted Link containing all the elements of both sorted Links $s$ & $t$.

![Diagram of linked list with elements 1, 5, 1, 4, 1, 1, 4, 5]

Do the same thing, but never call Link.

![Diagram of linked list with elements 1, 5, 1, 4]
### Linked List Exercises

Is a linked list \( s \) ordered from least to greatest?

\[
1 \rightarrow 3 \rightarrow 4 \hspace{1cm} 1 \rightarrow 4 \rightarrow 3
\]

Is a linked list \( s \) ordered from least to greatest by absolute value (or a key function)?

\[
1 \rightarrow -3 \rightarrow 4 \hspace{1cm} 1 \rightarrow 4 \rightarrow -3
\]

Create a sorted Link containing all the elements of both sorted Links \( s \) & \( t \).

\[
1 \rightarrow -3 \rightarrow 5 \hspace{1cm} 1 \rightarrow 4 \rightarrow 5
\]

Do the same thing, but never call Link.

\[
1 \xrightarrow{X} 5 \rightarrow 4
\]
Linked List Exercises

Is a linked list $s$ ordered from least to greatest?

Is a linked list $s$ ordered from least to greatest by absolute value (or a key function)?

Create a sorted Link containing all the elements of both sorted Links $s$ & $t$.

Do the same thing, but never call Link.