## Data Examples

Announcements

## Examples: Lists

## Lists in Environment Diagrams

## Lists in Environment Diagrams

Assume that before each example below we execute:
$\mathrm{s}=[2,3]$
$t=[5,6]$

## Lists in Environment Diagrams

Assume that before each example below we execute:
$\mathrm{s}=[2,3]$
$t=[5,6]$

## Operation

## Lists in Environment Diagrams

Assume that before each example below we execute:
$\mathrm{s}=[2,3]$
$t=[5,6]$

## Lists in Environment Diagrams

Assume that before each example below we execute:
$\mathrm{s}=[2,3]$
$t=[5,6]$

## Lists in Environment Diagrams

Assume that before each example below we execute:
$\mathrm{s}=[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |
| append adds one <br> element to a list |  |  |
|  |  |  |

## Lists in Environment Diagrams

Assume that before each example below we execute:
$\mathrm{s}=[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |
| append adds one <br> element to a list | S.append(t) <br> $t=0$ |  |
|  |  |  |

## Lists in Environment Diagrams

## Assume that before each example below we execute:

s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result | Global |
| :--- | :--- | :--- | :--- |
| append adds one <br> element to a list | s.append(t) <br> $\mathrm{t}=0$ |  |  |

## Lists in Environment Diagrams

```
Assume that before each example below we execute:
\(s=[2,3]\)
\(t=[5,6]\)
```

| Operation | Example | Result |
| :--- | :--- | :--- |
| append adds one <br> element to a list | s.append(t) <br> $t=0$ |  |
|  |  |  |
|  |  |  |



## Lists in Environment Diagrams

```
Assume that before each example below we execute:
\(\mathrm{s}=[2,3]\)
\(t=[5,6]\)
```

| Operation | Example | Result |
| :--- | :--- | :--- |
| append adds one <br> element to a list | S.append(t) <br> $\mathrm{t}=0$ |  |
|  |  |  |



## Lists in Environment Diagrams

```
Assume that before each example below we execute:
\(\mathrm{s}=[2,3]\)
\(t=[5,6]\)
```

| Operation | Example | Result |
| :--- | :--- | :--- |
| append adds one <br> element to a list | s.append(t) <br> t $=0$ |  |
|  |  |  |



## Lists in Environment Diagrams

```
Assume that before each example below we execute:
\(s=[2,3]\)
\(t=[5,6]\)
```



## Lists in Environment Diagrams

```
Assume that before each example below we execute:
\(\mathrm{s}=[2,3]\)
\(t=[5,6]\)
```

| Operation | Example | Result | Global |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append(t) } \\ & t=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  | list <br> 0 <br> 2 | ${ }^{1} 3$ |
|  |  |  |  | list |  |
|  |  |  |  | ${ }^{0} 5$ | $6$ |

## Lists in Environment Diagrams

```
Assume that before each example below we execute:
s = \([2,3]\)
\(t=[5,6]\)
```



## Lists in Environment Diagrams

```
Assume that before each example below we execute:
s = \([2,3]\)
\(t=[5,6]\)
```

| Operation | Example | Result |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append (t) } \\ & \mathrm{t}=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  | list <br> 0 <br> 2 | ${ }^{1} 3$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend }(t) \\ & t[1]=0 \end{aligned}$ |  |  | list |  |
|  |  |  |  | ${ }^{0} 5$ | ${ }^{1} 6$ |

## Lists in Environment Diagrams

```
Assume that before each example below we execute:
s = \([2,3]\)
\(t=[5,6]\)
```

| Operation | Example | Result | Global |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append }(\mathrm{t}) \\ & \mathrm{t}=0 \end{aligned}$ | $\begin{array}{ll} s \rightarrow[2,3,[5,6]] \\ t \rightarrow 0 \end{array}$ |  | $\begin{aligned} & \text { list } \\ & { }^{0} 2 \end{aligned}$ | ${ }^{1} 3$ | ${ }^{2} 5$ | ${ }^{3} 6$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend }(t) \\ & t[1]=0 \end{aligned}$ |  |  | list |  |  |  |
|  |  |  |  | ${ }^{0} 5$ | ${ }^{1} 6$ |  |  |

## Lists in Environment Diagrams

```
Assume that before each example below we execute:
s = \([2,3]\)
\(t=[5,6]\)
```

| Operation | Example | Result | Global |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append (t) } \\ & \mathrm{t}=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  | list <br> ${ }^{0} \mathbf{2}$ | ${ }^{1} 3$ | ${ }^{2} 5$ | ${ }^{3} 6$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend }(t) \\ & \mathrm{t}[1]=0 \end{aligned}$ |  |  | list |  |  |  |
|  |  |  |  |  | ${ }^{1} \times 0$ |  |  |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result | Global <br> s <br> t |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s. append }(t) \\ & t=0 \end{aligned}$ | $\begin{aligned} & \mathrm{s} \rightarrow[2,3,[5,6]] \\ & \mathrm{t} \rightarrow 0 \end{aligned}$ |  | list 0 | ${ }^{1} 3$ | ${ }^{2} 5$ | ${ }^{3} 6$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend }(\mathrm{t}) \\ & \mathrm{t}[1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  | t |  |  |  |
|  |  |  |  | ${ }^{0} 5$ | ${ }^{1} \times 0$ |  |  |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result | Global |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append (t) } \\ & \mathrm{t}=0 \end{aligned}$ | $\begin{aligned} & \mathrm{s} \rightarrow[2,3,[5,6]] \\ & \mathrm{t} \rightarrow 0 \end{aligned}$ |  | list <br> 0 <br> 2 | ${ }^{1} 3$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend }(t) \\ & \text { t[1] }=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  | list |  |
|  |  |  |  | ${ }^{0} 5$ | ${ }^{1} 6$ |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result | Global | list |
| :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append }(t) \\ & t=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  | 3 |
| ```extend adds all elements in one list to another list``` | $\begin{aligned} & \text { s.extend }(\mathrm{t}) \\ & \mathrm{t}[1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  |  |
| addition \& slicing create new lists containing existing elements | $\begin{aligned} & a=s+[t] \\ & b=a[1:] \\ & a[1]=9 \\ & b[1][1]=0 \end{aligned}$ |  |  | 6 |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result | Global |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append(t) } \\ & t=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  |  | $\begin{aligned} & \text { list } \\ & \begin{array}{\|c} 0 \\ 2 \end{array} \end{aligned}$ | $3$ |
| ```extend adds all elements in one list to another list``` | $\begin{aligned} & \text { s.extend }(\mathrm{t}) \\ & \mathrm{t}[1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  |  | list |  |
| addition \& slicing <br> create new lists containing existing elements | $\begin{aligned} & a=s+[t] \\ & b=a[1:] \\ & a[1]=9 \\ & b[1][1]=0 \end{aligned}$ |  |  |  | $\begin{array}{\|l} 0 \\ 5 \end{array}$ | ${ }^{1} 6$ |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> $\mathrm{s}=[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result | Global |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append }(\mathrm{t}) \\ & \mathrm{t}=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  | list 0 2 | ${ }^{1} 3$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend }(t) \\ & t[1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  |  |  |
| addition \& slicing create new lists containing existing elements | $\begin{aligned} & a=s+[t] \\ & b=a[1:] \\ & a[1]=9 \\ & b[1][1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3] \\ & t \rightarrow[5,0] \\ & a \rightarrow[2,9,[5,0]] \\ & b \rightarrow[3,[5,0]] \end{aligned}$ |  | $\begin{aligned} & \text { list } \\ & \hline 0 \end{aligned}$ |  |
| The list function also creates a new list containing existing elements | $\begin{aligned} & \mathrm{t}=\text { list(s) } \\ & \mathrm{s}[1]=0 \end{aligned}$ |  |  | $5$ | 6 |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> $\mathrm{s}=[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result | Global <br> s <br> $t$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append }(\mathrm{t}) \\ & \mathrm{t}=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  | ${ }^{\text {list }}$ | ${ }^{1} 3$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend }(t) \\ & \mathrm{t}[1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  | list |  |
| addition \& slicing create new lists containing existing elements | $\begin{aligned} & a=s+[t] \\ & b=a[1:] \\ & a[1]=9 \\ & b[1][1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3] \\ & t \rightarrow[5,0] \\ & a \rightarrow[2,9,[5,0]] \\ & b \rightarrow[3,[5,0]] \end{aligned}$ |  | 2 | 3 |
| The list function also creates a new list containing existing elements | $\begin{aligned} & \mathrm{t}=\text { list(s) } \\ & \mathrm{s}[1]=0 \end{aligned}$ |  |  | 5 | 6 |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append(t) } \\ & \mathrm{t}=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  | list ${ }^{0} 2$ | ${ }^{1} 3$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend( } \mathrm{t}) \\ & \mathrm{t}[1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  | lis |  |
| addition \& slicing create new lists containing existing elements | $\begin{aligned} & a=s+[t] \\ & b=a[1:] \\ & a[1]=9 \\ & b[1][1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3] \\ & t \rightarrow[5,0] \\ & a \rightarrow[2,9,[5,0]] \\ & b \rightarrow[3,[5,0]] \end{aligned}$ |  | 2 | 3 |
| The list function also creates a new list containing existing elements | $\begin{aligned} & \mathrm{t}=\text { list(s) } \\ & \mathrm{s}[1]=0 \end{aligned}$ |  |  | 5 | 6 |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result | $\begin{array}{c\|c\|c} \hline \text { Global } \\ s & - \\ t & - \end{array}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append }(\mathrm{t}) \\ & \mathrm{t}=0 \end{aligned}$ | $\operatorname{sic}_{\substack{s \\ t \rightarrow 0}}[2,[5,6]]$ |  | list <br> 0 <br> 2 | ${ }^{1} 80$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s. extend }(t) \\ & \mathrm{t}[1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  | list |  |
| addition \& slicing create new lists containing existing elements | $\begin{aligned} & a=s+[t] \\ & b=a[1:] \\ & a[1]=9 \\ & b[1][1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3] \\ & t \rightarrow[5,0] \\ & a \rightarrow[2,9,[5,0]] \\ & b \rightarrow[3,[5,0]] \end{aligned}$ |  | 2 | 3 |
| The list function also creates a new list containing existing elements | $\begin{aligned} & \mathrm{t}=\operatorname{list}(\mathrm{s}) \\ & \mathrm{s}[1]=0 \end{aligned}$ |  |  | 5 | 6 |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> $\mathrm{s}=[2,3]$ <br> $t=[5,6]$

| Operation | Example | Result |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| append adds one element to a list | $\begin{aligned} & \text { s.append(t) } \\ & \mathrm{t}=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,[5,6]] \\ & t \rightarrow 0 \end{aligned}$ |  | ${ }^{\text {list }}$ | ${ }^{1} \times 0$ |
| extend adds all elements in one list to another list | $\begin{aligned} & \text { s.extend( } \mathrm{t}) \\ & \mathrm{t}[1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3,5,6] \\ & t \rightarrow[5,0] \end{aligned}$ |  | list |  |
| addition \& slicing create new lists containing existing elements | $\begin{aligned} & a=s+[t] \\ & b=a[1:] \\ & a[1]=9 \\ & b[1][1]=0 \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3] \\ & t \rightarrow[5,0] \\ & a \rightarrow[2,9,[5,0]] \\ & b \rightarrow[3,[5,0]] \end{aligned}$ |  | 2 | 3 |
| The list function also creates a new list containing existing elements | $\begin{aligned} & \mathrm{t}=\text { list(s) } \\ & \mathrm{s}[1]=0 \end{aligned}$ | $\begin{array}{lll} s \rightarrow[2, & 0] \\ t \rightarrow[2, & 3] \end{array}$ |  | 5 | 6 |

## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

## Assume that before each example below we execute: <br> s = $[2,3]$ <br> $t=[5,6]$



## Lists in Environment Diagrams

Assume that before each example below we execute:
$\mathrm{s}=[2,3]$
$t=[5,6]$

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$
Operation
Example
Result

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |

pop removes \& returns
the last element

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |
| pop removes \& returns <br> the last element | $\mathrm{t}=\mathrm{s.pop()}$ |  |
|  |  |  |
|  |  |  |

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |
| pop removes \& returns <br> the last element | $t=s . p o p()$ | $s \rightarrow[2]$ <br> $t \rightarrow 3$ |

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |
| pop removes \& returns <br> the last element | $\mathrm{t}=\mathrm{s} . \mathrm{pop}()$ | $\mathrm{s} \rightarrow$ [2] <br> $\mathrm{t} \rightarrow 3$ |

remove removes the
first element equal
to the argument

## Lists in Environment Diagrams

## Assume that before each example below we execute:

s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |
| pop removes \& returns <br> the last element | t = s.pop() | s $\rightarrow$ [2] <br> $t \rightarrow 3$ |
| remove removes the <br> first element equal <br> to the argument | t.extend(t) <br> t.remove(5) |  |
|  |  |  |
|  |  |  |

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |
| pop removes \& returns <br> the last element | $\mathrm{t}=\mathrm{s.pop()}$ | $\mathrm{s} \rightarrow$ [2] <br> $\mathrm{t} \rightarrow 3$ |
| remove removes the <br> first element equal <br> to the argument | t.extend(t) <br> t.remove(5) | $\mathrm{s} \rightarrow[2, ~ 3]$ <br> $\mathrm{t} \rightarrow[6,5,6]$ |

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :--- | :--- | :--- |
| pop removes \& returns <br> the last element | $\mathrm{t}=\mathrm{s.pop()}$ | $\mathrm{s} \rightarrow$ [2] <br> $\mathrm{t} \rightarrow$ 3 |
| remove removes the <br> first element equal <br> to the argument | t.extend(t) <br> $\mathrm{t} . \operatorname{remove(5)}$ | $\mathrm{s} \rightarrow$ [2, 3] <br> $\mathrm{t} \rightarrow[6,5,6]$ |
| slice assignment can <br> remove elements from <br> a list by assigning <br> [] to a slice. |  |  |

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :---: | :---: | :---: |
| pop removes \& returns the last element | t = s.pop() | $\begin{aligned} & s \rightarrow[2] \\ & t \rightarrow 3 \end{aligned}$ |
| remove removes the first element equal to the argument | $\begin{aligned} & \text { t.extend(t) } \\ & \text { t. remove(5) } \end{aligned}$ | $\begin{aligned} & \mathrm{s} \rightarrow[2,3] \\ & \mathrm{t} \rightarrow[6,5,6] \end{aligned}$ |
| slice assignment can remove elements from a list by assigning [] to a slice. | $\begin{aligned} & s[: 1]=[] \\ & \mathrm{t}[0: 2]=[] \end{aligned}$ |  |

## Lists in Environment Diagrams

Assume that before each example below we execute:
s = $[2,3]$
$t=[5,6]$

| Operation | Example | Result |
| :---: | :---: | :---: |
| pop removes \& returns the last element | t = s.pop() | $\begin{aligned} & s \rightarrow[2] \\ & t \rightarrow 3 \end{aligned}$ |
| remove removes the first element equal to the argument | $\begin{aligned} & \text { t.extend(t) } \\ & \text { t. remove(5) } \end{aligned}$ | $\begin{aligned} & s \rightarrow[2,3] \\ & t \rightarrow[6,5,6] \end{aligned}$ |
| slice assignment can remove elements from a list by assigning [] to a slice. | $\begin{aligned} & s[: 1]=[] \\ & t[0: 2]=[] \end{aligned}$ | $\begin{array}{ll} s \rightarrow[3] \\ t \rightarrow[] \end{array}$ |

Lists in Lists in Lists in Environment Diagrams

```
t = [1, 2, 3]
t[1:3] = [t]
t.extend(t)
```

$t=[[1,2],[3,4]]$
t[0]. append(t[1:2])

Lists in Lists in Lists in Environment Diagrams

$$
\begin{aligned}
& t=[1,2,3] \\
& t[1: 3]=[t] \\
& t . \text { extend }(t)
\end{aligned}
$$


$t=[[1,2],[3,4]]$
t[0]. append(t[1:2])

Lists in Lists in Lists in Environment Diagrams

$$
\begin{aligned}
& \mathrm{t}=[1,2,3] \\
& \mathrm{t}[1: 3]=[\mathrm{t}] \\
& \mathrm{t} . \text { extend( } \mathrm{t})
\end{aligned}
$$


[t] evaluates to: 0
$t=[[1,2],[3,4]]$
t[0]. append(t[1:2])

Lists in Lists in Lists in Environment Diagrams

$$
\begin{aligned}
& t=[1,2,3] \\
& t[1: 3]=[t] \\
& \text { t.extend }(\mathrm{t})
\end{aligned}
$$


$t=[[1,2],[3,4]]$
t[0]. append(t[1:2])

Lists in Lists in Lists in Environment Diagrams

$$
\begin{aligned}
& t=[1,2,3] \\
& t[1: 3]=[t] \\
& \text { t.extend }(\mathrm{t})
\end{aligned}
$$


$t=[[1,2],[3,4]]$
t[0]. append(t[1:2])

Lists in Lists in Lists in Environment Diagrams

$$
\begin{aligned}
& t=[1,2,3] \\
& t[1: 3]=[t] \\
& t . \text { extend }(t)
\end{aligned}
$$


$t=[[1,2],[3,4]]$
t[0]. append(t[1:2])

Lists in Lists in Lists in Environment Diagrams

$$
\begin{aligned}
& t=[1,2,3] \\
& t[1: 3]=[t] \\
& \text { t.extend }(\mathrm{t})
\end{aligned}
$$


$t=[[1,2],[3,4]]$
t[0]. append(t[1:2])

Lists in Lists in Lists in Environment Diagrams


$$
[1,[. . .], 1,[. . .]]
$$

```
t = [[1, 2], [3, 4]]
t[0].append(t[1:2])
```


## Lists in Lists in Lists in Environment Diagrams

```
t = [1, 2, 3]
t[1:3] = [t]
t.extend(t)
```


[1, [...], 1, [...]]
$t=[[1,2],[3,4]]$ t[0]. append(t[1:2])


| list |  |
| :---: | :---: |
| $\mathbf{1}$ |  |

## Lists in Lists in Lists in Environment Diagrams

```
t = [1, 2, 3]
t[1:3] = [t]
t.extend(t)
```


[1, [...], 1, [...]]
$t=[[1,2],[3,4]]$ t[0]. append(t[1:2])


## Lists in Lists in Lists in Environment Diagrams

```
t = [1, 2, 3]
t[1:3] = [t]
t.extend(t)
```


[1, [...], 1, [...]]
$t=[[1,2],[3,4]]$ t[0]. append(t[1:2])


## Lists in Lists in Lists in Environment Diagrams

```
t = [1, 2, 3]
t[1:3] = [t]
t.extend(t)
```


[1, [...], 1, [...]]
$t=[[1,2],[3,4]]$ t[0]. append(t[1:2])

[[1, 2, [[3, 4]]], [3, 4]]

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
class Worker:
greeting = 'Sir'
def __init__(self):
self.elf = 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'

## 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

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

## 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()
>>> jack
>>> 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'
greeting =
def work(self): print(Worker.work(self)) return 'I gather wealth'
jack = Worker()
jack.greeting = 'Maam'
>>> Worker().work()
>>> jack
>>> jack.work()
>>> john.work()
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
jack <Worker>

>>> 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()
>>> jack
>>> jack.work()
>>> john.work()
>>> john.elf.work(john)
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
jack <Worker>

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()
>>> jack
>>> jack.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 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()
>>> jack
>>> jack.work()
>>> john.work()
>>> john.elf.work(john)
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
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()
>>> Worker().work()
'Sir, I work'
>>> jack
>>> jack.work()
>>> john.work()
>>> john.elf.work(john)
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
jack <Worker>

john <Bourgeoisie>

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()
'Sir, I work'
>>> jack
>>> jack.work()
>>> john.work()
>>> john.elf.work(john)
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
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'
```

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)
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
<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'
```

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)
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
<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.work()
'Maam, I work'
>>> john.work()
>>> john.elf.work(john)
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
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'
>>> jack
Peon
>>> jack.work()
'Maam, I work'
>>> john.work()
>>> john.elf.work(john)
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
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'
>>> 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()
```

>>> 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.work()
'Maam, I work'
>>> john.work()
Peon, I work
'I gather wealth'
>>> john.elf.work(john)
'Peon, I work'
<class Worker>
greeting: 'Sir'
<class Bourgeoisie>
greeting: 'Peon'
jack <Worker>
elf:
greeting: 'Maam'
john <Bourgeoisie>
elf:


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?
$\left[\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4] \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}\right]$

## Using Built-In Functions \& Comprehensions

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

$$
\left.\begin{array}{cccccc}
{[-4,} & -3, & -2, & 3, & 2, & 4] \\
0 & 1 & 2 & 3 & 4 & 5
\end{array}\right\rangle[2,4]
$$

## Using Built-In Functions \& Comprehensions

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

$$
\left.\begin{array}{cccccc}
{[-4,} & -3, & -2, & 3, & 2, & 4] \\
0 & 1 & 2 & 3 & 4 & 5
\end{array}\right\rangle[2,4] \quad[1,2,3,4,5]
$$

## Using Built-In Functions \& Comprehensions

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

$$
\left.\left.\begin{array}{cccccc}
{[-4,} & -3, & -2, & 3, & 2, & 4 \\
0 & 1 & 2 & 3 & 4 & 5
\end{array}\right\rangle[2,4] \quad[1,2,3,4,5]\right\rangle[0]
$$

## Using Built-In Functions \& Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?
$\left.\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4] \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}\right\rangle[2,4]$
$[1,2,3,4,5]$
[0]

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

## Using Built-In Functions \& Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?
$\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4 \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}$
$[2,4]$
$[1,2,3,4,5]$
[0]

What's the largest sum of two adjacent elements in a list s? (Assume len(s) > 1)
$[-4,-3,-2,3,2,4]$

## Using Built-In Functions \& Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?
$\left[\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4] \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}\right.$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

$$
[-4,-3,-2,3,2,4]\rangle 6
$$

## Using Built-In Functions \& Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?
$\left[\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4] \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}\right.$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

$$
[-4,-3,-2,3,2,4]\rangle 6 \quad[-4,3,-2,-3,2,-4]
$$

## Using Built-In Functions \& Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?
$\left[\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4] \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}\right.$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

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

## Using Built-In Functions \& Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?
$\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4 \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

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

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?
$\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4 \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

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

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?
$\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4 \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

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

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]
```


## Using Built-In Functions \& Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?
$\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4 \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

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

Create a dictionary mapping each digit d to the lists of elements in s that end with d . $[5,8,13,21,34,55,89]\rangle\{1:[21], 3:[13], 4:[34], 5:[5,55], 8:[8], 9:[89]\}$

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?
$\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4 \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

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

Create a dictionary mapping each digit d to the lists of elements in s that end with d .
$[5,8,13,21,34,55,89]\rangle\{1:[21], 3:[13], 4:[34], 5:[5,55], 8:[8], 9:[89]\}$
Does every element equal some other element in s?

$$
[-4,-3,-2,3,2,4]\rangle \text { False }
$$

## Using Built-In Functions \& Comprehensions

What are the indices of all elements in a list $s$ that have the smallest absolute value?
$\begin{array}{cccccc}{[-4,} & -3, & -2, & 3, & 2, & 4 \\ 0 & 1 & 2 & 3 & 4 & 5\end{array}$
$[2,4]$
$[1,2,3,4,5]$
[0]

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

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

Create a dictionary mapping each digit $d$ to the lists of elements in $s$ that end with $d$.
$[5,8,13,21,34,55,89]\rangle\{1:[21], 3:[13], 4:[34], 5:[5,55], 8:[8], 9:[89]\}$

Does every element equal some other element in s?

$$
[-4,-3,-2,3,2,4]\rangle \text { False } \quad[4,3,2,3,2,4]\rangle \text { True }
$$

## Examples: Linked Lists

Linked List Exercises

Linked List Exercises

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 Exercises

Is a linked list s ordered from least to greatest?


## 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)?

## 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)?


## 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)?


## 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?


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?


3

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?


3

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.

## Linked List Exercises

Is a linked list s ordered from least to greatest?


3

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.


## Linked List Exercises

Is a linked list s ordered from least to greatest?


3

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.


## 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.


