

Python

Slicing

Lists, strings, and tuples are all *sequences*

Lists, strings, and tuples are all *sequences*

Can be indexed by integers in the range $0 \dots \text{len}(X) - 1$

Lists, strings, and tuples are all *sequences*

Can be indexed by integers in the range $0 \dots \text{len}(X) - 1$

Can also be *sliced* using a range of indices

Lists, strings, and tuples are all *sequences*

Can be indexed by integers in the range $0 \dots \text{len}(X) - 1$

Can also be *sliced* using a range of indices

```
>>> element = 'uranium'  
>>>
```

0	1	2	3	4	5	6	7
u	r	a	n	i	u	m	
-	-	-	-	-	-	-	
7	6	5	4	3	2	1	

Lists, strings, and tuples are all *sequences*

Can be indexed by integers in the range $0 \dots \text{len}(X) - 1$

Can also be *sliced* using a range of indices

```
>>> element = 'uranium'
```

```
>>> print(element[1:4])
```

ran

```
>>>
```

0	1	2	3	4	5	6	7
u	r	a	n	i	u	m	
-	-	-	-	-	-	-	
7	6	5	4	3	2	1	

Lists, strings, and tuples are all *sequences*

Can be indexed by integers in the range $0 \dots \text{len}(X) - 1$

Can also be *sliced* using a range of indices

```
>>> element = 'uranium'
```

```
>>> print(element[1:4])
```

```
uran
```

```
>>> print(element[:4])
```

```
uran
```

```
>>>
```

0	1	2	3	4	5	6	7
u	r	a	n	i	u	m	
-	-	-	-	-	-	-	
7	6	5	4	3	2	1	

Lists, strings, and tuples are all *sequences*

Can be indexed by integers in the range $0 \dots \text{len}(X) - 1$

Can also be *sliced* using a range of indices

```
>>> element = 'uranium'
```

```
>>> print(element[1:4])
```

```
ran
```

```
>>> print(element[:4])
```

```
uran
```

```
>>> print(element[4:])
```

```
ium
```

```
>>>
```

0	1	2	3	4	5	6	7
u	r	a	n	i	u	m	
-	-	-	-	-	-	-	
7	6	5	4	3	2	1	

Lists, strings, and tuples are all *sequences*

Can be indexed by integers in the range $0 \dots \text{len}(X) - 1$

Can also be *sliced* using a range of indices

```
>>> element = 'uranium'
```

```
>>> print(element[1:4])
```

```
ran
```

```
>>> print(element[:4])
```

```
uran
```

```
>>> print(element[4:])
```

```
ium
```

```
>>> print(element[-4:])
```

```
nium
```

```
>>>
```

0	1	2	3	4	5	6	7
u	r	a	n	i	u	m	
-	-	-	-	-	-	-	
7	6	5	4	3	2	1	

Python checks bounds when indexing

Python checks bounds when indexing

But truncates when slicing

Python checks bounds when indexing

But truncates when slicing

```
>>> element = 'uranium'  
>>>
```

0	1	2	3	4	5	6	7
u	r	a	n	i	u	m	
-	-	-	-	-	-	-	
7	6	5	4	3	2	1	

Python checks bounds when indexing

But truncates when slicing

```
>>> element = 'uranium'  
>>> print(element[400])
```

IndexError: string index out of range

```
>>>
```

0	1	2	3	4	5	6	7
u	r	a	n	i	u	m	
-	-	-	-	-	-	-	
7	6	5	4	3	2	1	0

Python checks bounds when indexing

But truncates when slicing

```
>>> element = 'uranium'
```

```
>>> print(element[400])
```

IndexError: string index out of range

```
>>> print(element[1:400])
```

uranium

```
>>>
```

0 1 2 3 4 5 6 7

u	r	a	n	i	u	m	
---	---	---	---	---	---	---	--

- - - - -
7 6 5 4 3 2 1

So `text[1:3]` is 0, 1, or 2 characters long

So `text[1:3]` is 0, 1, or 2 characters long

`' '`

`' '`

`'a'`

`' '`

`'ab'`

`'b'`

`'abc'`

`'bc'`

`'abcdef'`

`'bc'`

Slicing always creates a new collection

Slicing always creates a new collection

Beware of aliasing

Slicing always creates a new collection

Beware of aliasing

```
>>> points = [[10, 10], [20, 20], [30, 30], [40, 40]]  
>>>
```

Slicing always creates a new collection

Beware of aliasing

```
>>> points = [[10, 10], [20, 20], [30, 30], [40, 40]]  
>>> middle = points[1:-1]  
>>>
```

Slicing always creates a new collection

Beware of aliasing

```
>>> points = [[10, 10], [20, 20], [30, 30], [40, 40]]
>>> middle = points[1:-1]
>>> middle[0][0] = 'whoops'
>>>
```

Slicing always creates a new collection

Beware of aliasing

```
>>> points = [[10, 10], [20, 20], [30, 30], [40, 40]]
>>> middle = points[1:-1]
>>> middle[0][0] = 'whoops'
>>> middle[1][0] = 'aliasing'
>>>
```

Slicing always creates a new collection

Beware of aliasing

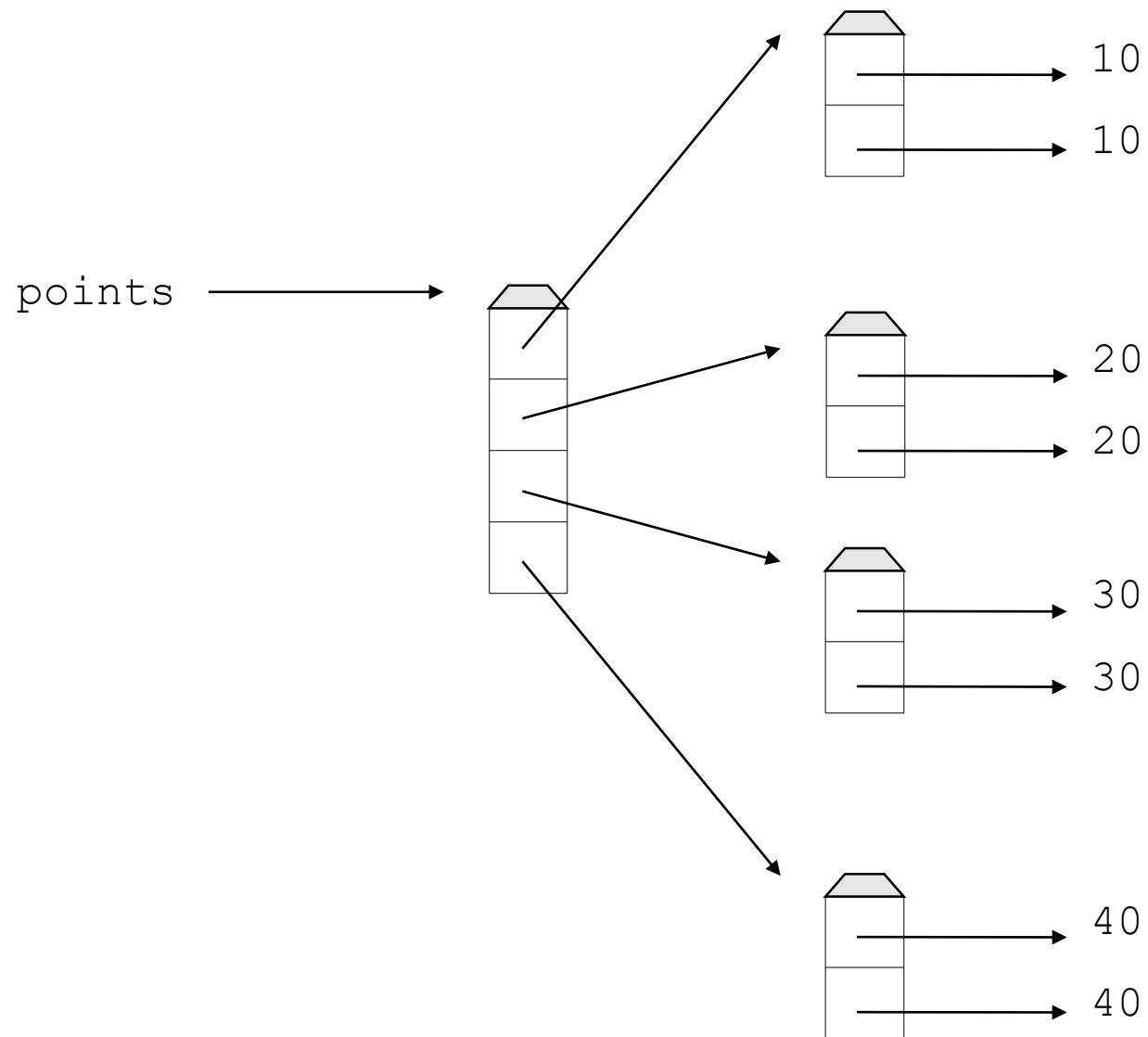
```
>>> points = [[10, 10], [20, 20], [30, 30], [40, 40]]
>>> middle = points[1:-1]
>>> middle[0][0] = 'whoops'
>>> middle[1][0] = 'aliasing'
>>> print(middle)
[['whoops', 20], ['aliasing', 30]]
>>>
```

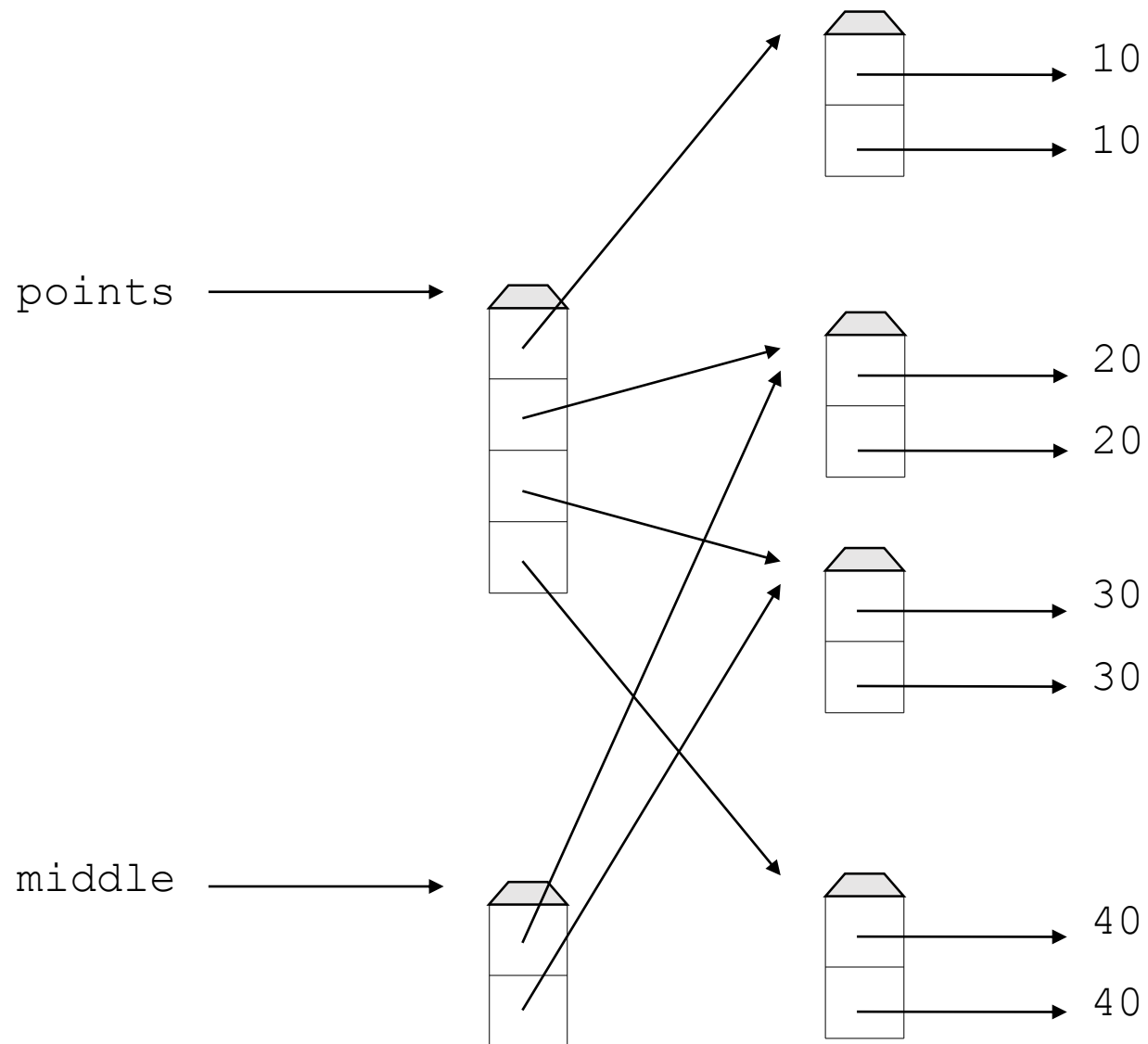
Slicing always creates a new collection

Beware of aliasing

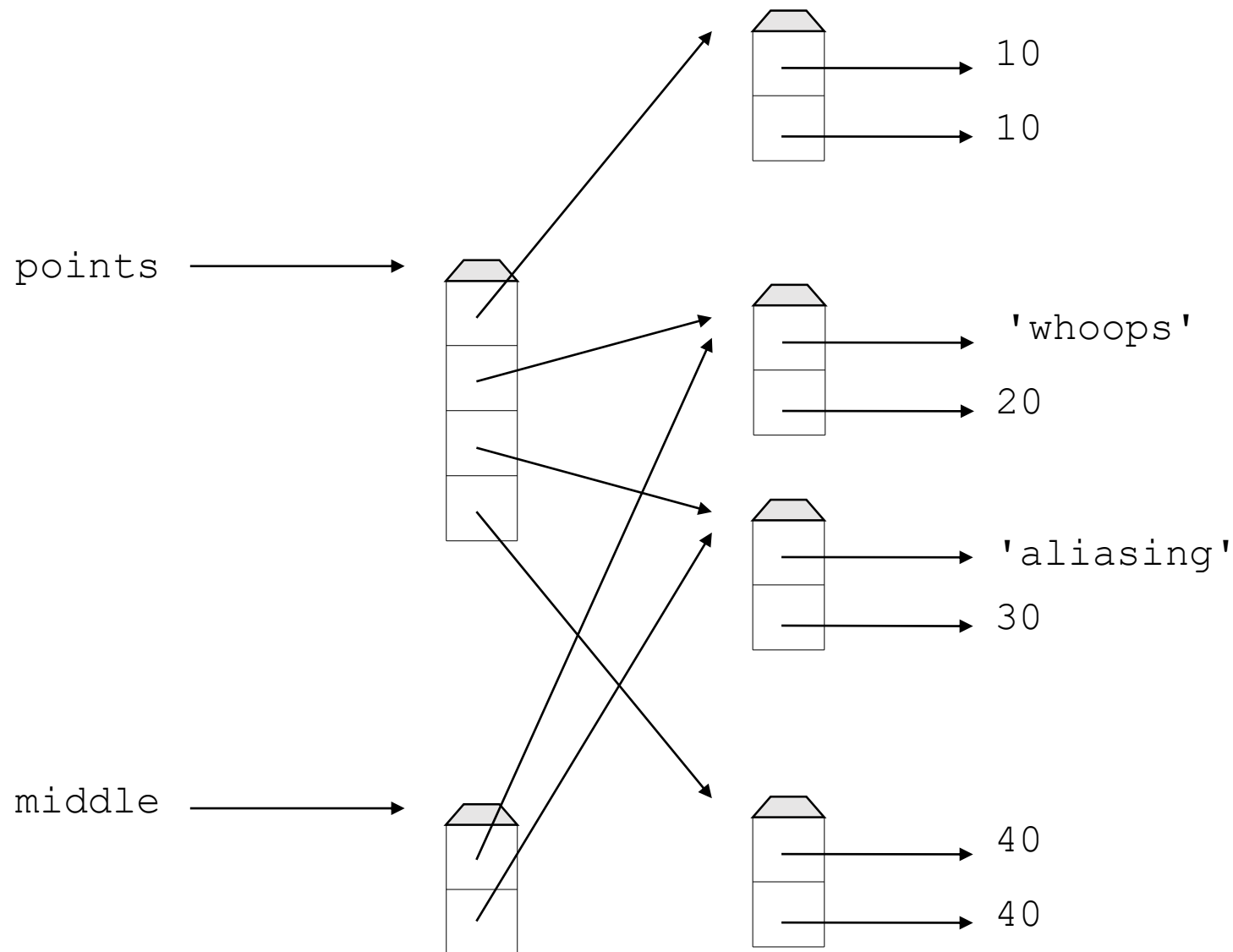
```
>>> points = [[10, 10], [20, 20], [30, 30], [40, 40]]
>>> middle = points[1:-1]
>>> middle[0][0] = 'whoops'
>>> middle[1][0] = 'aliasing'
>>> print(middle)
[['whoops', 20], ['aliasing', 30]]
>>> print(points)
[[10, 10], ['whoops', 20], ['aliasing', 30], [40, 40]]
>>>
```


STOP HERE











created by

Greg Wilson

October 2010



Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

See <http://software-carpentry.org/license.html> for more information.