Python

Lists

Loops let us do things many times

Loops let us do things many times

Collections let us store many values together

Loops let us do things many times

Collections let us store many values together

Most popular collection is a *list*

Create using `[value, value, ...]`

Create using `[value, value, ...]`

Get/set values using `var[index]`

Create using `[value, value, ...]`

Get/set values using `var[index]`

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases
['He', 'Ne', 'Ar', 'Kr']
```

Create using `[value, value, ...]`

Get/set values using `var[index]`

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases
['He', 'Ne', 'Ar', 'Kr']

print gases[1]
Ne
```

Index from 0, not 1

Index from 0, not 1

Reasons made sense for C in 1970...

Index from 0, not 1

Reasons made sense for C in 1970...

It's an error to try to access out of range

Index from 0, not 1

Reasons made sense for C in 1970...

It's an error to try to access out of range

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases[4]
IndexError: list index out of range
```

Use `len(list)` to get length of list

Use `len(list)` to get length of list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print len(gases)
4
```

Use `len(list)` to get length of list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print len(gases)
4
```

Returns 0 for the *empty list*

```
etheric = []
print len(etheric)
0
```

Some negative indices work

Some negative indices work

`values[-1]` is last element, `values[-2]` next-to-last, ...

Some negative indices work

`values[-1]` is last element, `values[-2]` next-to-last, ...

```
gases = ['He', 'Ne', 'Ar', 'Kr']
```

Some negative indices work

`values[-1]` is last element, `values[-2]` next-to-last, ...

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases[-1], gases[-4]
Kr He
```

Some negative indices work

`values[-1]` is last element, `values[-2]` next-to-last, ...

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases[-1], gases[-4]
Kr He
```

`values[-1]` is much nicer than `values[len(values)-1]`

Some negative indices work

`values[-1]` is last element, `values[-2]` next-to-last, ...

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases[-1], gases[-4]
Kr He
```

`values[-1]` is much ~~nicer~~ than `values[len(values)-1]`

less error prone

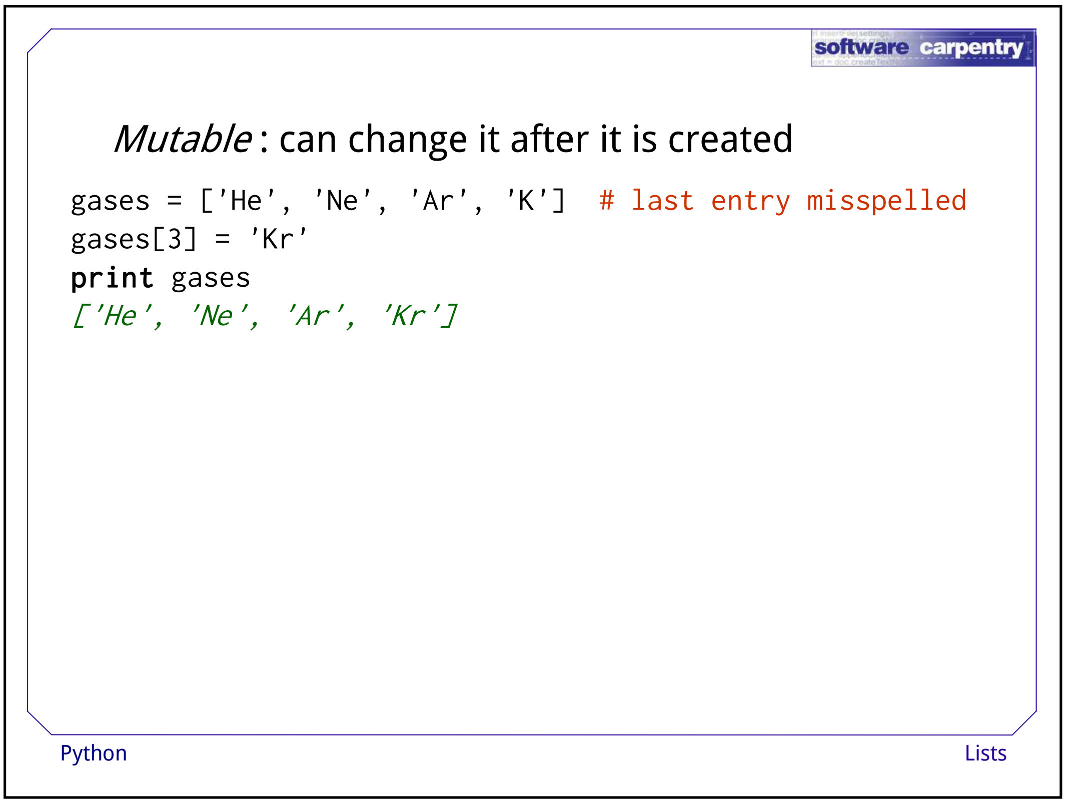
Mutable : can change it after it is created

Mutable : can change it after it is created

```
gases = ['He', 'Ne', 'Ar', 'K']  # last entry misspelled
```

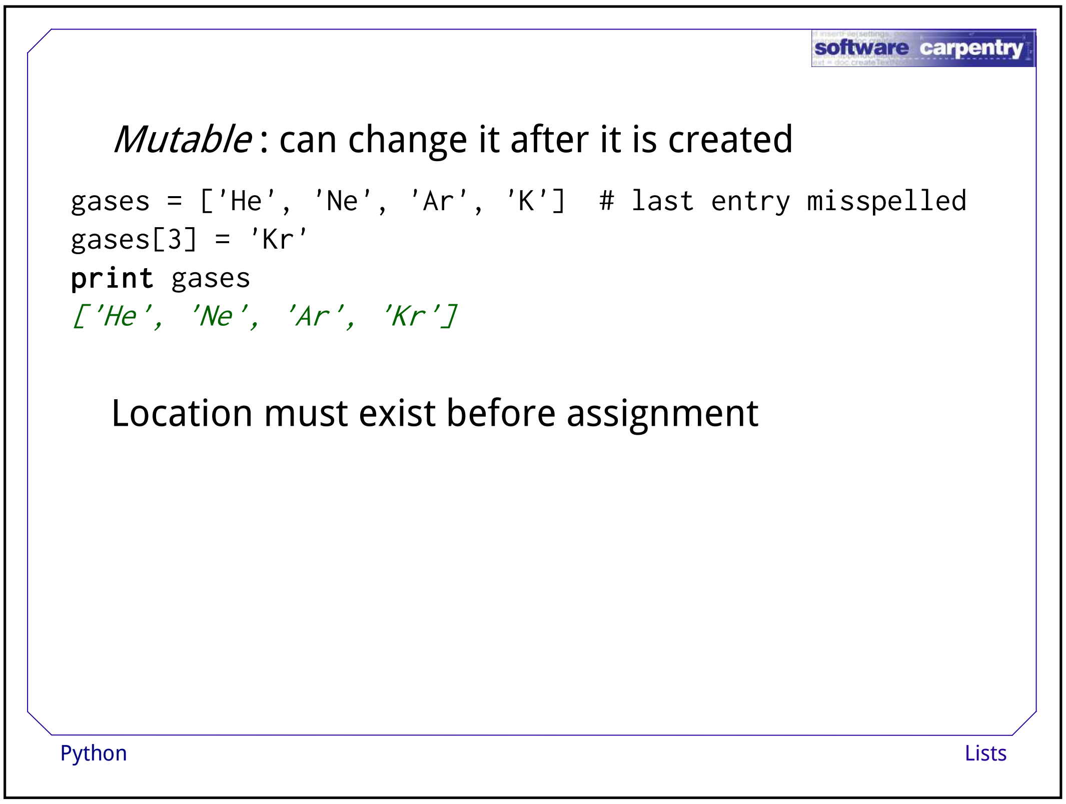
Mutable : can change it after it is created

```
gases = ['He', 'Ne', 'Ar', 'K']  # last entry misspelled
gases[3] = 'Kr'
```



Mutable : can change it after it is created

```
gases = ['He', 'Ne', 'Ar', 'K']  # last entry misspelled
gases[3] = 'Kr'
print gases
['He', 'Ne', 'Ar', 'Kr']
```



Mutable : can change it after it is created

```
gases = ['He', 'Ne', 'Ar', 'K']  # last entry misspelled
gases[3] = 'Kr'
print gases
['He', 'Ne', 'Ar', 'Kr']
```

Location must exist before assignment

Mutable : can change it after it is created

```
gases = ['He', 'Ne', 'Ar', 'K']  # last entry misspelled
gases[3] = 'Kr'
print gases
['He', 'Ne', 'Ar', 'Kr']
```

Location must exist before assignment

```
gases = ['He', 'Ne', 'Ar', 'Kr']
```

Mutable : can change it after it is created

```
gases = ['He', 'Ne', 'Ar', 'K']  # last entry misspelled
gases[3] = 'Kr'
print gases
['He', 'Ne', 'Ar', 'Kr']
```

Location must exist before assignment

```
gases = ['He', 'Ne', 'Ar', 'Kr']
gases[4] = 'Xe'
IndexError: list assignment index out of range
```

Heterogeneous : can store values of many kinds

Heterogeneous : can store values of many kinds

```
helium = ['He', 2]
neon = ['Ne', 8]
```

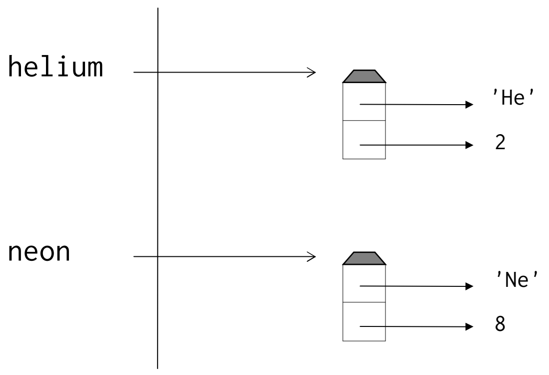
Heterogeneous : can store values of many kinds

```
helium = ['He', 2]
neon = ['Ne', 8]
```

[string, int]

Heterogeneous : can store values of many kinds

```
helium = ['He', 2]
neon = ['Ne', 8]
```

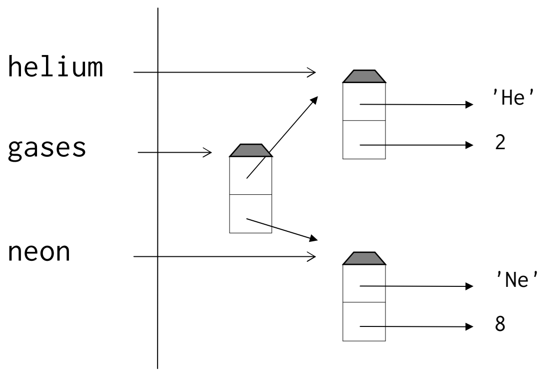


Heterogeneous : can store values of many kinds

```
helium = ['He', 2]
neon = ['Ne', 8]
gases = [helium, neon]
```

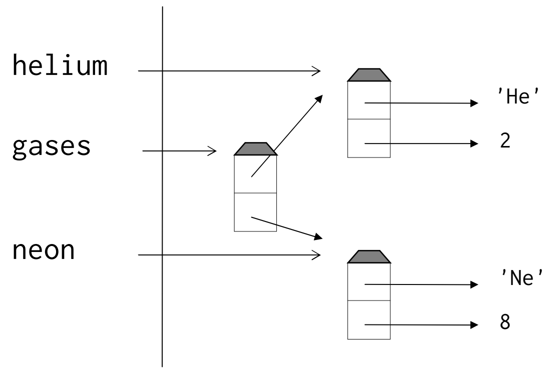
Heterogeneous : can store values of many kinds

```
helium = ['He', 2]
neon = ['Ne', 8]
gases = [helium, neon]
```



Heterogeneous : can store values of many kinds

```
helium = ['He', 2]
neon = ['Ne', 8]
gases = [helium, neon]
```



Devote a whole episode to this

Loop over elements to "do all"

Loop over elements to "do all"

Use `while` to step through all possible indices

Loop over elements to "do all"

Use `while` to step through all possible indices

```
gases = ['He', 'Ne', 'Ar', 'Kr']
i = 0
while i < len(gases):
    print gases[i]
    i += 1
```

software carpentry

Loop over elements to "do all"

Use `while` to step through all possible indices

```
gases = ['He', 'Ne', 'Ar', 'Kr']
i = 0
while i < len(gases):
    print gases[i]
    i += 1
```

First legal index

software carpentry

Loop over elements to "do all"

Use `while` to step through all possible indices

```
gases = ['He', 'Ne', 'Ar', 'Kr']
i = 0
while i < len(gases):
    print gases[i]
    i += 1
```

Next index

Loop over elements to "do all"

Use `while` to step through all possible indices

```
gases = ['He', 'Ne', 'Ar', 'Kr']
i = 0
while i < len(gases):
    print gases[i]
    i += 1
```

Defines set of legal indices

Loop over elements to "do all"

Use `while` to step through all possible indices

```
gases = ['He', 'Ne', 'Ar', 'Kr']
i = 0
while i < len(gases):
    print gases[i]
    i += 1
He
Ne
Ar
Kr
```

Loop over elements to "do all"

Use `while` to step through all possible indices

```
gases = ['He', 'Ne', 'Ar', 'Kr']
i = 0
while i < len(gases):
    print gases[i]
    i += 1
He
Ne
Ar
Kr
```

Tedious to type in over and over again

Loop over elements to "do all"

Use `while` to step through all possible indices

```
gases = ['He', 'Ne', 'Ar', 'Kr']
i = 0
while i < len(gases):
    print gases[i]
    i += 1
He
Ne
Ar
Kr
```

Tedious to type in over and over again

And it's easy to forget the "+= 1" at the end

Use a `for` loop to access each value in turn

Use a `for` loop to access each value in turn

```
gases = ['He', 'Ne', 'Ar', 'Kr']
for gas in gases:
    print gas
He
Ne
Ar
Kr
```

Use a `for` loop to access each value in turn

```
gases = ['He', 'Ne', 'Ar', 'Kr']
for gas in gases:
    print gas
He
Ne
Ar
Kr
```

Loop variable assigned each *value* in turn

Use a `for` loop to access each value in turn

```
gases = ['He', 'Ne', 'Ar', 'Kr']
for gas in gases:
    print gas
He
Ne
Ar
Kr
```

Loop variable assigned each *value* in turn

Not each index

Use a `for` loop to access each value in turn

```
gases = ['He', 'Ne', 'Ar', 'Kr']
for gas in gases:
    print gas
He
Ne
Ar
Kr
```

Loop variable assigned each *value* in turn

Not each index

Because that's the most common case

Can delete entries entirely (shortens the list)

Can delete entries entirely (shortens the list)

```
gases = ['He', 'Ne', 'Ar', 'Kr']
```

Can delete entries entirely (shortens the list)

```
gases = ['He', 'Ne', 'Ar', 'Kr']
del gases[0]
```

Can delete entries entirely (shortens the list)

```
gases = ['He', 'Ne', 'Ar', 'Kr']
del gases[0]
print gases
['Ne', 'Ar', 'Kr']
```

Can delete entries entirely (shortens the list)

```
gases = ['He', 'Ne', 'Ar', 'Kr']
del gases[0]
print gases
['Ne', 'Ar', 'Kr']
del gases[2]
```

Python Lists

Can delete entries entirely (shortens the list)

```
gases = ['He', 'Ne', 'Ar', 'Kr']
del gases[0]
print gases
['Ne', 'Ar', 'Kr']
del gases[2]
print gases
['Ne', 'Ar']
```

Can delete entries entirely (shortens the list)

```
gases = ['He', 'Ne', 'Ar', 'Kr']
del gases[0]
print gases
['Ne', 'Ar', 'Kr']
del gases[2]
print gases
['Ne', 'Ar']
```

Yes, deleting an index that doesn't exist is an error

Appending values to a list lengthens it

Appending values to a list lengthens it

```
gases = []
```

Appending values to a list lengthens it

```
gases = []
gases.append('He')
```

Appending values to a list lengthens it

```
gases = []
gases.append('He')
gases.append('Ne')
```

Appending values to a list lengthens it

```
gases = []
gases.append('He')
gases.append('Ne')
gases.append('Ar')
```

Appending values to a list lengthens it

```
gases = []
gases.append('He')
gases.append('Ne')
gases.append('Ar')
print gases
['He', 'Ne', 'Ar']
```

Appending values to a list lengthens it

```
gases = []
gases.append('He')
gases.append('Ne')
gases.append('Ar')
print gases
['He', 'Ne', 'Ar']
```

Most operations on lists are *methods*

Appending values to a list lengthens it

```
gases = []
gases.append('He')
gases.append('Ne')
gases.append('Ar')
print gases
['He', 'Ne', 'Ar']
```

Most operations on lists are *methods*

A function that belongs to (and usually operates on) specific data

Appending values to a list lengthens it

```
gases = []
gases.append('He')
gases.append('Ne')
gases.append('Ar')
print gases
['He', 'Ne', 'Ar']
```

Most operations on lists are *methods*

A function that belongs to (and usually operates on) specific data

```
thing . method (args)
```

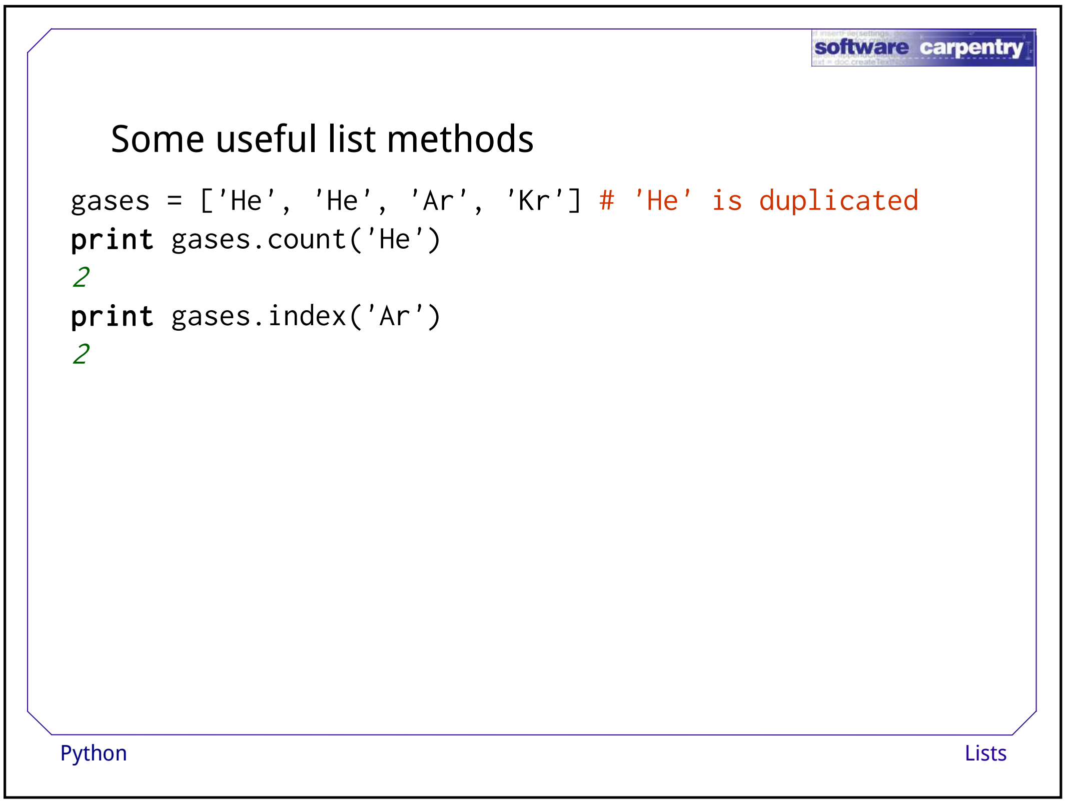
Some useful list methods

Some useful list methods

```
gases = ['He', 'He', 'Ar', 'Kr'] # 'He' is duplicated
```

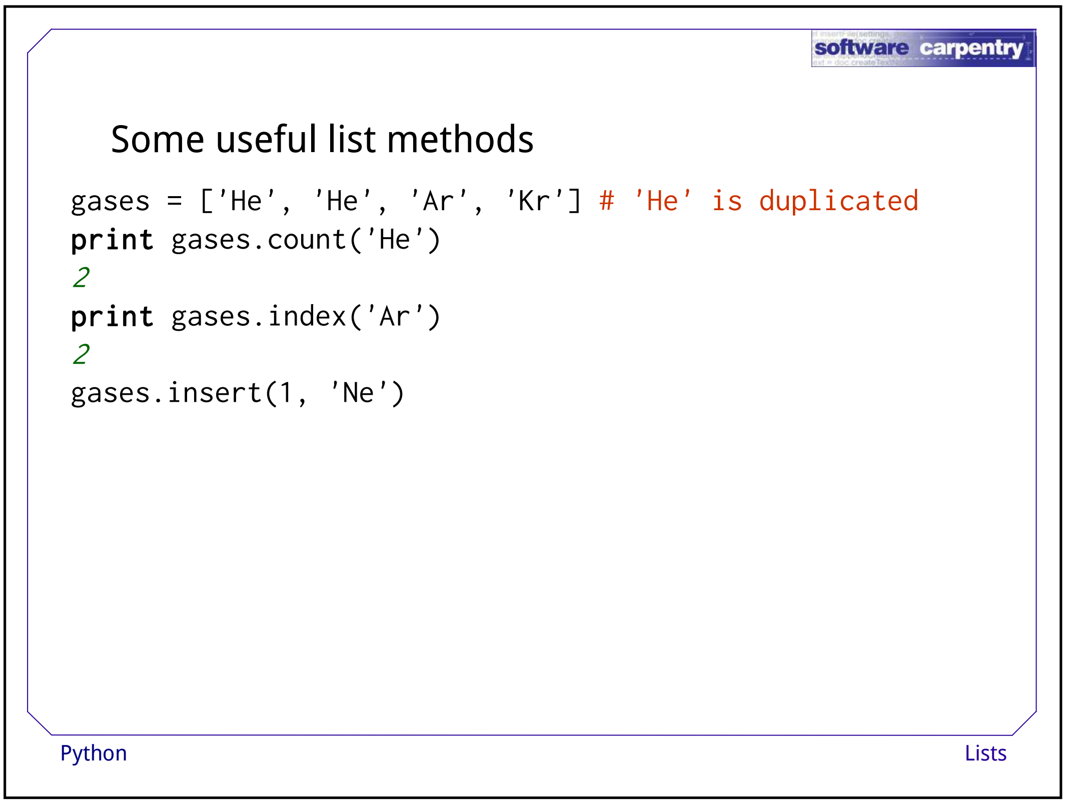
Some useful list methods

```
gases = ['He', 'He', 'Ar', 'Kr'] # 'He' is duplicated
print gases.count('He')
2
```



Some useful list methods

```
gases = ['He', 'He', 'Ar', 'Kr'] # 'He' is duplicated
print gases.count('He')
2
print gases.index('Ar')
2
```



Some useful list methods

```
gases = ['He', 'He', 'Ar', 'Kr'] # 'He' is duplicated
print gases.count('He')
2
print gases.index('Ar')
2
gases.insert(1, 'Ne')
```

Some useful list methods

```
gases = ['He', 'He', 'Ar', 'Kr'] # 'He' is duplicated
print gases.count('He')
2
print gases.index('Ar')
2
gases.insert(1, 'Ne')
print gases
['He', 'Ne', 'He', 'Ar', 'Kr']
```

Two that are often used incorrectly

Two that are often used incorrectly

```
gases = ['He', 'Ne', 'Ar', 'Kr']
```

Two that are often used incorrectly

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases.sort()
None
```

Two that are often used incorrectly

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases.sort()
None
print gases
['Ar', 'He', 'Kr', 'Ne']
```

Two that are often used incorrectly

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases.sort()
None
print gases
['Ar', 'He', 'Kr', 'Ne']
print gases.reverse()
None
```

Two that are often used incorrectly

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases.sort()
None
print gases
['Ar', 'He', 'Kr', 'Ne']
print gases.reverse()
None
print gases
['Ne', 'Kr', 'He', 'Ar']
```

Two that are often used incorrectly

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases.sort()
None
print gases
['Ar', 'He', 'Kr', 'Ne']
print gases.reverse()
None
print gases
['Ne', 'Kr', 'He', 'Ar']
```

A common bug

Two that are often used incorrectly

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print gases.sort()
None
print gases
['Ar', 'He', 'Kr', 'Ne']
print gases.reverse()
None
print gases
['Ne', 'Kr', 'He', 'Ar']
```

A common bug

`gases = gases.sort()` assigns `None` to `gases`

Use `in` to test for membership

Use `in` to test for membership

```
gases = ['He', 'Ne', 'Ar', 'Kr']
```

Use `in` to test for membership

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print 'He' in gases
True
```

Use `in` to test for membership

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print 'He' in gases
True
if 'Pu' in gases:
    print 'But plutonium is not a gas!'
else:
    print 'The universe is well ordered.'
```

Use `in` to test for membership

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print 'He' in gases
True
if 'Pu' in gases:
    print 'But plutonium is not a gas!'
else:
    print 'The universe is well ordered.'
The universe is well ordered.
```

Use `range` to construct lists of numbers

Use `range` to construct lists of numbers

```
print range(5)
[0, 1, 2, 3, 4]
```

Use range to construct lists of numbers

```
print range(5)
[0, 1, 2, 3, 4]
print range(2, 6)
[2, 3, 4, 5]
```

Use range to construct lists of numbers

```
print range(5)
[0, 1, 2, 3, 4]
print range(2, 6)
[2, 3, 4, 5]
print range(0, 10, 3)
[0, 3, 6, 9]
```

Use range to construct lists of numbers

```
print range(5)
[0, 1, 2, 3, 4]
print range(2, 6)
[2, 3, 4, 5]
print range(0, 10, 3)
[0, 3, 6, 9]
print range(10, 0)
[]
```

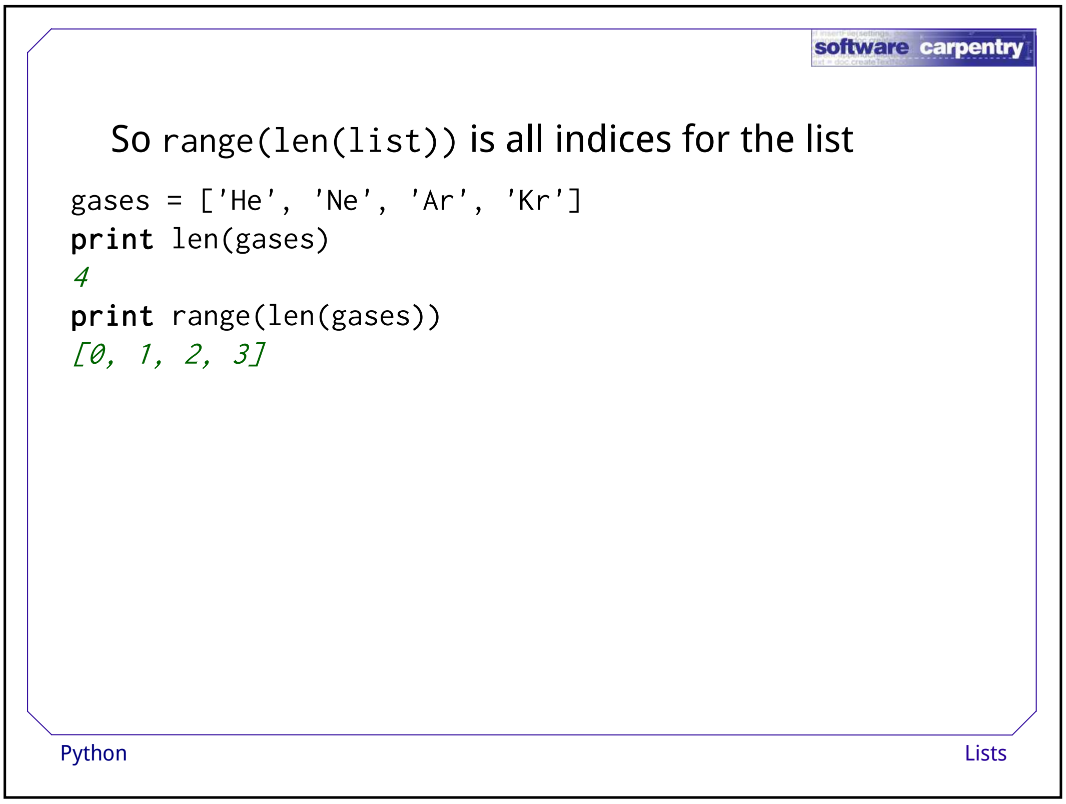
So range(len(list)) is all indices for the list

So `range(len(list))` is all indices for the list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
```

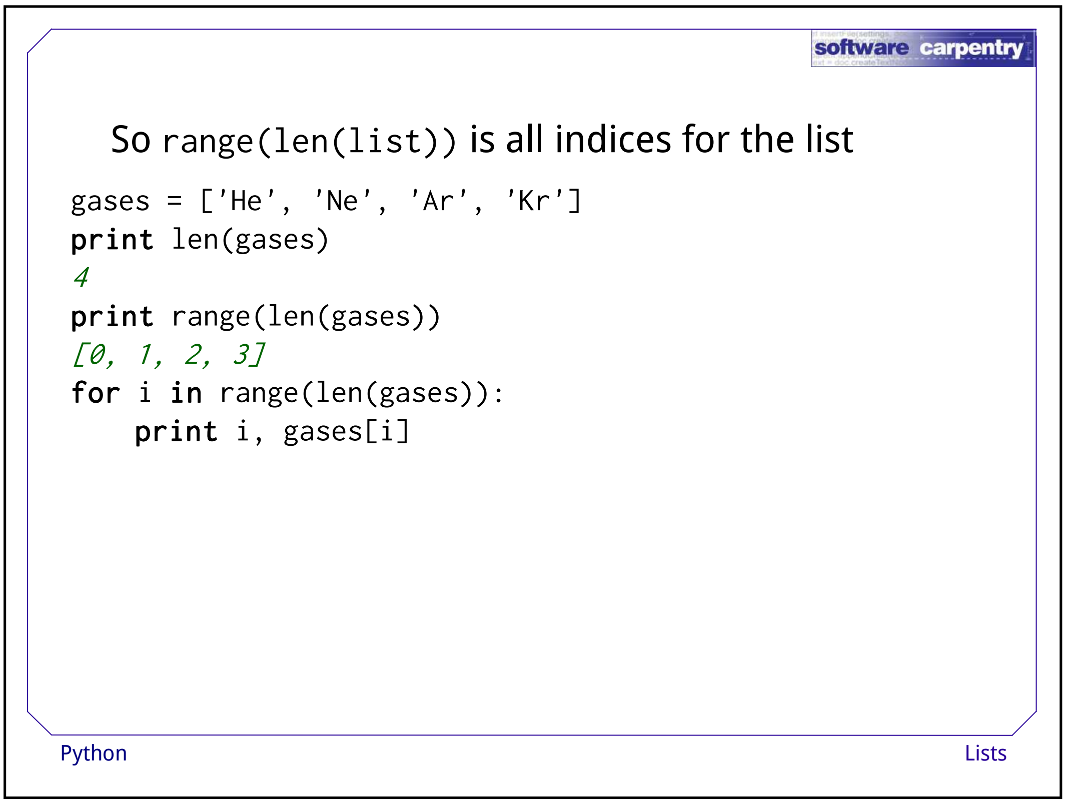
So `range(len(list))` is all indices for the list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print len(gases)
4
```



So `range(len(list))` is all indices for the list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print len(gases)
4
print range(len(gases))
[0, 1, 2, 3]
```



So `range(len(list))` is all indices for the list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print len(gases)
4
print range(len(gases))
[0, 1, 2, 3]
for i in range(len(gases)):
    print i, gases[i]
```

So `range(len(list))` is all indices for the list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print len(gases)
4
print range(len(gases))
[0, 1, 2, 3]
for i in range(len(gases)):
    print i, gases[i]
0 He
1 Ne
2 Ar
3 Kr
```

So `range(len(list))` is all indices for the list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print len(gases)
4
print range(len(gases))
[0, 1, 2, 3]
for i in range(len(gases)):
    print i, gases[i]
0 He
1 Ne
2 Ar
3 Kr
```

A very common *idiom* in Python

narrated by

Dominique Vuvan

October 2010