Code Appetizer

Make a list called `movies` that contains your three favorite movies.

How would you access the second movie?

Can you think of another way to access the second movie?

Code Appetizer

Make a list called `movies` that contains your three favorite movies.

```
movies = [“The Notebook”, “Wall-E”, “The Social Network”]
```

How would you access the second movie?

Hackbright Academy

Code Appetizer

Make a list called `movies` that contains your three favorite movies.

```
movies = ["The Notebook", "Wall-E", "The Social Network"]
```

How would you access the second movie?

```
movies[1] or
movies[-2]
```

Agenda

6:30 - 6:40 Code Appetizer
6:40 – 6:45 Rapid Review
6:45 – 6:55 Less Rapid Review
6:55 – 7:10 Lists, Part 2
7:10 – 8:00 Lists Exercise 1/2
8:00 – 8:10 Break
8:10 – 8:25 For Loops Lecture
8:25 – 9:00 For Loops Exercise
9:00 – 9:20 Hackbright Bart Simulator
9:20 – 9:25 Exit Ticket

Hackbright Academy

Lists, Part 2

Hackbright Academy

First, A Review

- Creating a list

```
greetings = [“hi”,“hello”,“hey”]
```

- Accessing an element in a list

```
greetings         ⇒    [“hi”,“hello”,“hey”]
greetings[0:2]    ⇒    [“hi”,“hello”]
greetings[0]      ⇒    “hi”
greetings[-1]     ⇒    “hey”
```

Hackbright Academy

Changing Values in a List

Lists are *mutable*, so you can change the value of any of the items in a list.

```
letters = [‘A’, ‘B’, ‘C’]
letters[0] = ‘z’
print letters ⇒  [‘z’, ‘B’, ‘C’]
```

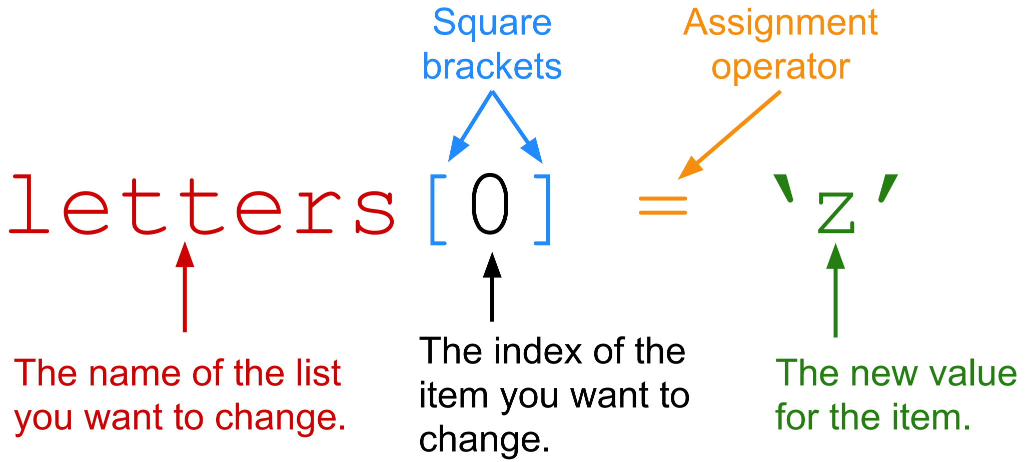
Hackbright Academy

Changing Values in a List

You can only *reassign* the value of an item in a list if there is already an item there.

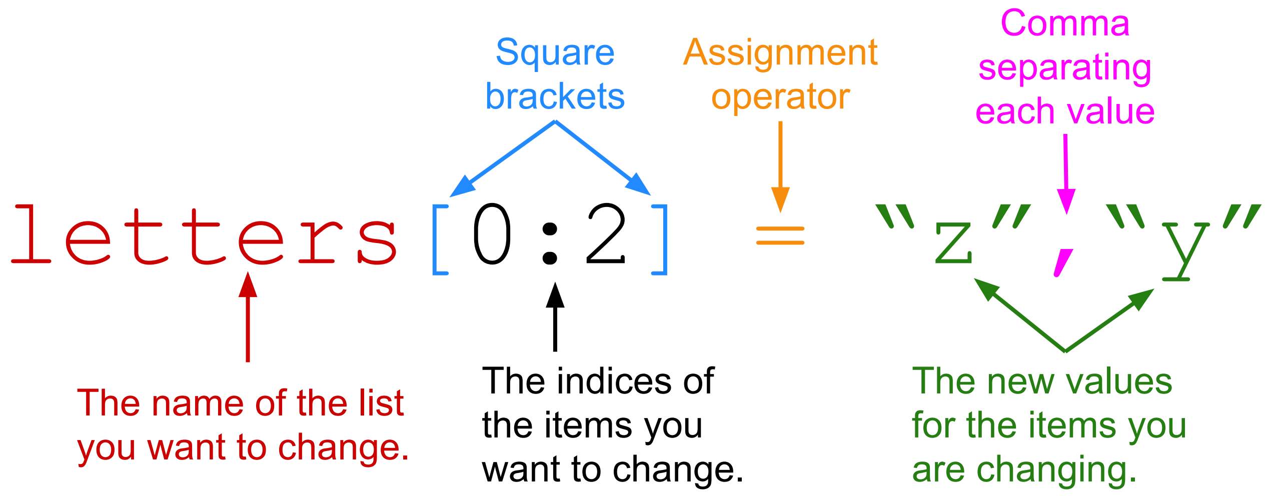
```
empty_list = [ ]
empty_list[0] = “hi”
   ⇒ IndexError: list assignment
     index out of range
```

Changing Values in a List - Recipe


Square brackets
Assignment operator
The name of the list you want to change.
The index of the item you want to change.
The new value for the item.

```
letters[0] = 'z'
```

Changing Values in a List - Recipe



```
letters[0:2] = "z", "y"
```

Square brackets
Assignment operator
Comma separating each value
The name of the list you want to change.
The indices of the items you want to change.
The new values for the items you are changing.

Hackbright Academy

Changing Values in a List - Slicing

```
letters = [‘A’, ‘B’, ‘C’]
letters[0:2] = ‘z’, ‘y’
print letters ⇒  [‘z’, ‘y’, ‘C’]
```

Hackbright Academy

Adding an Item to a List

Here is our list:

```
evens = [2,4]
```

We want to add the numbers `6` and `8` to the end of the list, so that we get this:

```
print evens    ⇒   [2,4,6,8]
```

Hackbright Academy

Adding an Item to a List

Using *list concatenation*

```
evens = [2,4]
evens = evens + [6, 8]
```

The items ***must*** be a list, even if there is only 1 item.

concatenation operator

Since this process **doesn't** change the original `evens`, you have to *reassign* `evens` to this new list

The list you want to add the items to.

Items you want to add to the list.

```
print evens
```

$\Rightarrow$ `[2,4,6,8]`

Hackbright Academy

Adding an Item to a List

Using the `append` *method*

```
evens = [2,4]
evens.append(6)
```

The name of the method you are calling. It is just like calling a function!

```
evens.append(8)
```

The list you want to add the item to.

Tells the computer that you are using a method

The item you want to add to the list

```
print evens     ⇒     [2,4,6,8]
```

Hackbright Academy

Adding an Item to a List

Using the `extend` *method*

```
evens = [2,4]
```

The name of the method you are calling. It is just like calling a function!

```
evens.extend([6,8])
```

The list you want to add the item to.

Tells the computer that you are using a method

The item(s) you want to add to the list. **They must be in the form of a list.**

```
print evens
```

⇒

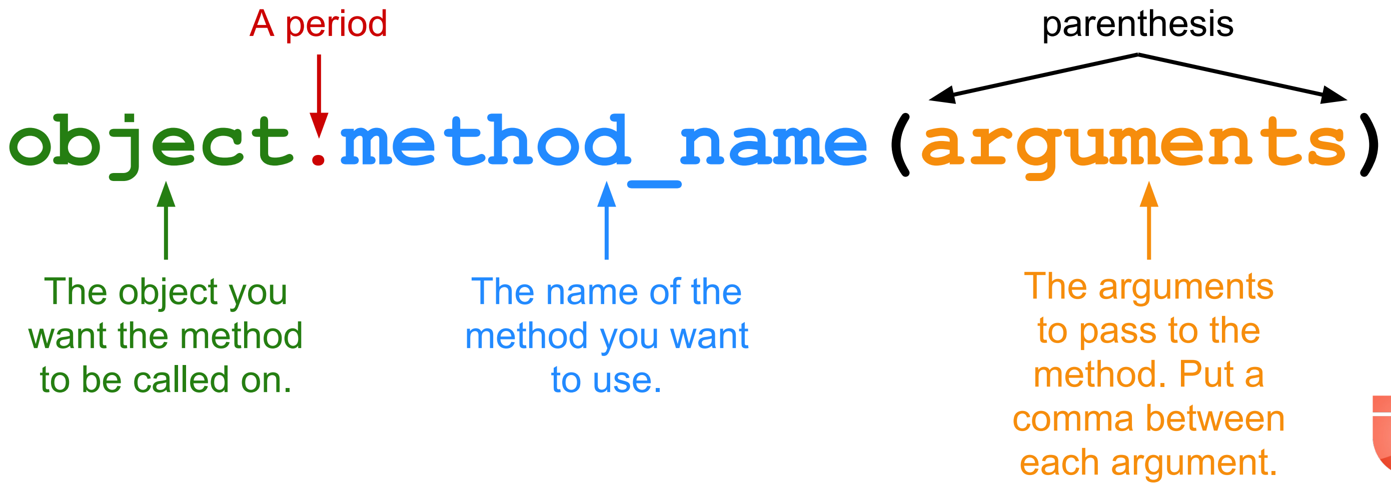
```
[2,4,6,8]
```

Hackbright Academy

Aside - Methods

A *method* is a function that acts on an object.

The recipe for *calling* a method is:



```
object.method_name(arguments)
```

A period
parenthesis
The object you want the method to be called on.
The name of the method you want to use.
The arguments to pass to the method. Put a comma between each argument.

Hackbright Academy

Aside - Why/when use a method?

When you want to perform a *specific action* on a *specific object*.

Adding “blue” to an existing list of colors

```
colors.append(“blue”)
```

Hackbright Academy

Removing an Item from a List

Here is our list:

```
evens = [2,4,6,8]
```

We want to remove the number 8 from the list:

```
print evens    ⇒   [2,4,6]
```

Hackbright Academy

Removing an Item from a List

Using the `del` statement

```
evens = [2,4,6,8]
```

The index of the item you want to remove from the list.

```
del evens[3]
```

Tells python that you want to delete an element from the list

Tells the computer what list you want to delete from

```
print evens     ⇒     [2,4,6]
```

Hackbright Academy

Removing an Item from a List

Using the `del` statement

```
evens = [2,4,6,8]

del evens[3]
```

The index of the item you want to remove from the list.

What's another way we could write this to get the last element of the list?

Tells python that you want to delete an element from the list

Tells the computer what list you want to delete from

```
print evens      ⇒      [2,4,6]
```

Hackbright Academy

Removing an Item from a List

Using the `del` statement

```
evens = [2,4,6,8]
```

The index of the item you want to remove from the list.

What's another way we could write this to get the last element of the list?

```
del evens[len(evens)-1]
```

```
del evens[3]
```

Tells python that you want to delete an element from the list

Tells the computer what list you want to delete from

```
print evens    ⇒    [2,4,6]
```

Hackbright Academy

Removing an Item from a List

Using the `del` statement

```
evens = [2,4,6,8]

del evens[3]
```

The index of the item you want to remove from the list.

Tells python that you want to delete an element from the list

Tells the computer what list you want to delete from

```
print evens
```

⇒ `[2,4,6]`

What's another way we could write this to get the last element of the list?

```
del evens[len(evens)-1]
del evens[-1]
```

Hackbright Academy

Removing an Item from a List

Using the `remove` method

```
evens = [2,4,6,8]

evens.remove(8)
```

The item you want to remove from the list. **Note: NOT the index!! If there are two 8's, it will remove the first 8.**

Tells the computer what list you want to delete from

Tells the computer you want to use a method

The name of the method you want to call. It's just like calling a function!

```
print evens    ⇒    [2,4,6]
```

Hackbright Academy

Removing an Item from a List

Using the `pop` method

```
evens = [2,4,6,8]
```

The pop method ***always*** removes the last item from the list, so we don't have to give it an index!

```
evens.pop()
```

Tells the computer what list you want to delete from

Tells the computer you want to use a method

The name of the method you want to call. It's just like calling a function!

```
print evens    ⇒    [2,4,6]
```

Hackbright Academy

Exercise Time!

1. Do List Exercises 2.
2. Finish List Exercises 1.
   a. Flag me down when you are done so I can check it.

Hackbright Academy