## Chapter 4. Python Lists

```
import numpy as np

me = 9.11e-31      # mass of electron
c  = 299792458     # speed of light

u  = 0.1 * c       # particle velocity

gamma = 1 / np.sqrt(1-(u/c)**2)   # gamma factor

KE = (gamma-1) * me * c**2        # relativistic kinetic energy
```

# Python for Physicists

# Python Lists

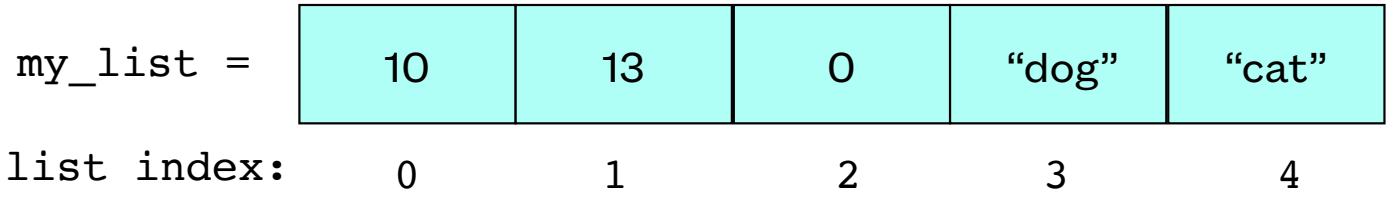
- A Python list is a container that can hold a collection of items, like numbers, words, or even other lists, all in a specific order.
- You can think of a list like a row of boxes, each with a number (the index) so you can look inside, replace what's there, or add new boxes at the end.


- Use square brackets to create a list:

```
my_list = [10, 13, 0, “dog”, “cat”]
```

# List indexing

- The boxes (or elements) in a list are numbered, starting from 0.
- The box number is called the **index**. You can think of the index as the item's address.



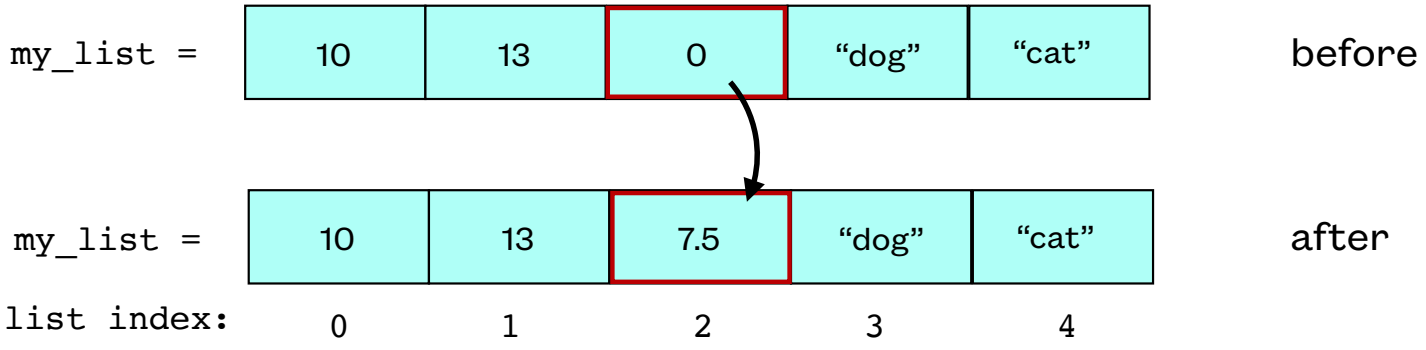

- You can **Fetch** an item by placing the index in square brackets:

```
first_item  = my_list[0]          →  10
second_item = my_list[1]          →  13
last_item   = my_list[-1]         →  "cat"
next_to_last_item = my_list[-2]   →  "dog"
```

# Modifying List elements

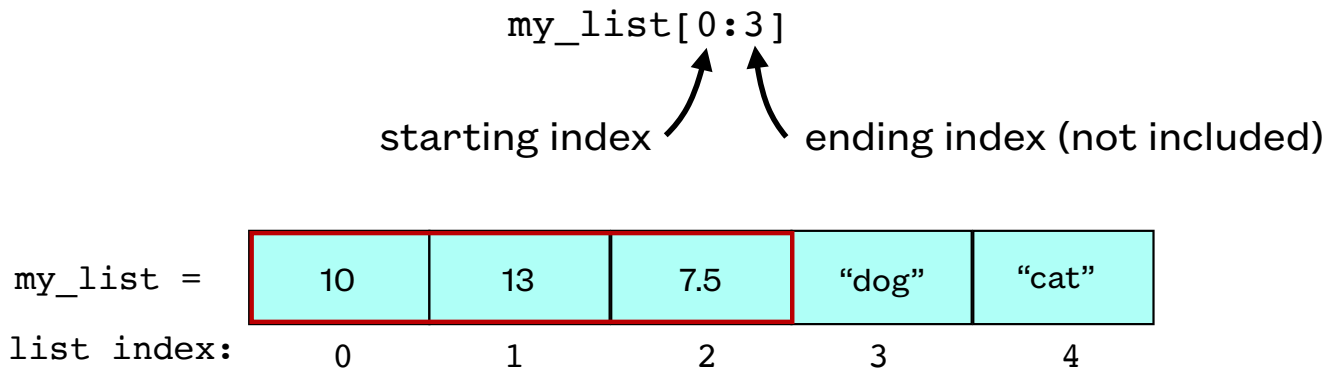
- You can replace the contents of a “box” by assigning it a new value:

```
my_list[2] = 7.5
```



# Slicing

- You can also fetch more than one box at a time using the colon notation:




- The elements are fetched from the starting index up to, but not including the ending index

```
my_list[0:3]  →  [10, 13, 7.5]
```

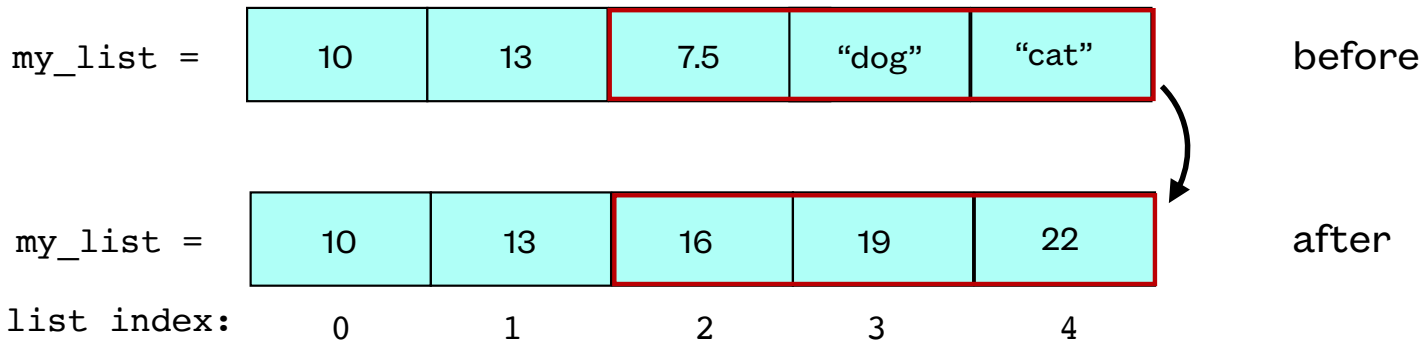
- If you leave the starting or ending index off, Python defaults to the beginning or end of the list:

```
my_list[:3]  →  [10, 13, 7.5]          my_list[3:]  →  ["dog", "cat"]
```

# Modifying multiple elements with slicing

- You can also assign more than one box at a time using slicing:

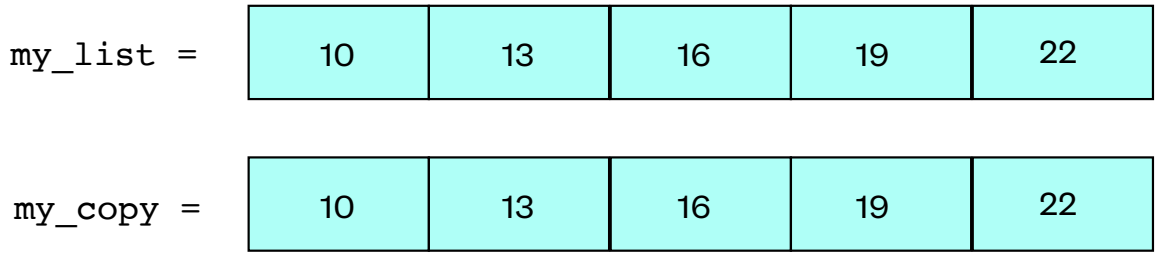
```
my_list[2:] = [16, 19, 22]
```



# Copying Lists

- If you need to copy a list, you'll want to use the .copy() method.

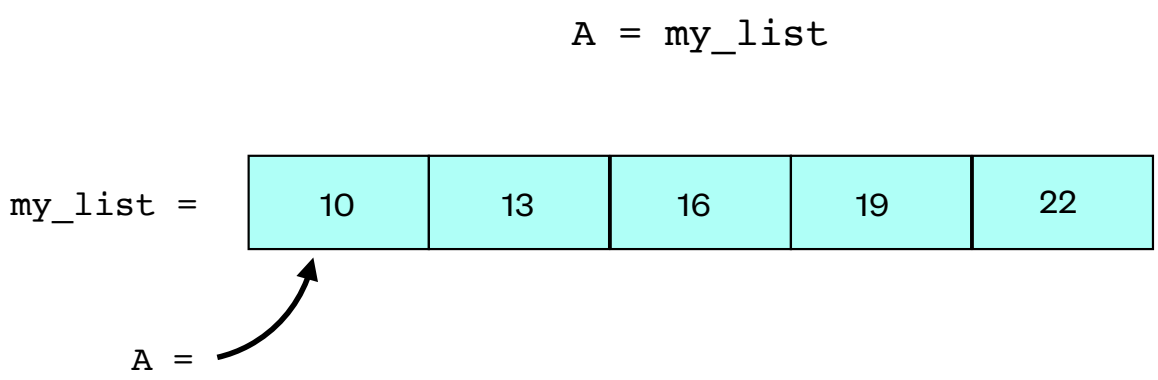
```
my_copy = my_list.copy()
```




- If you make a change to one, the other won't be affected

# Creating an Alias

- If you simply assign your list to another variable, such as A, Python does **not** make a copy




- The variable A will be an alias that points to the same “boxes” (i.e. memory locations) as my_list.
- Thus changing `my_list` will also change the contents of `A`

# Common list methods:

```
A.copy(x)      # creates a copy of A
A.append(x)    # appends value x to end of A
A.extend(B)    # appends list B to end of A
A.remove(30)   # deletes first element in A whose value = 30
A.insert(n,x)  # insert value x at index n in list A
A.count(x)     # number of occurrences of x in A
A.sort()       # sorts items in list A
A.reverse(x)   # reverse order of items in A
```

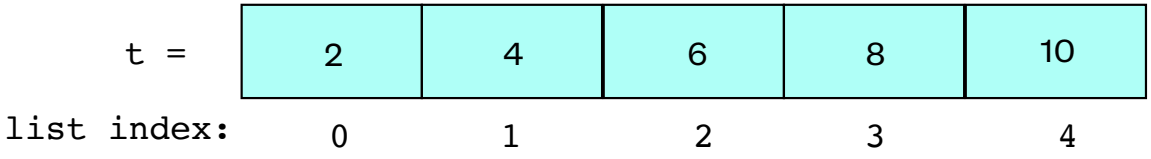
# Common functions that act on lists:

```
len(A)          # number of items in list A
sum(A)          # sum of items in list A
min(A)          # minimum value of items in A
max(A)          # maximum value of items in A

A[0],A[1] = A[1],A[0]   # swap items 0 and 1

A = []          # sets A to the empty list
del A[3]        # deletes index=3 item from list
```

# Creating an Alias

- If you simply assign your list to another variable, such as A, Python does **not** make a copy

```
A = my_list
```




- The variable A will be an alias that points to the same “boxes” (i.e. memory locations) as my_list.
- Thus changing `my_list` will also change the contents of `A`