PYTHON FOR DATA SCIENCE

UNIT 09

LISTS

LIST BASICS

I. WHAT ARE LISTS?

- Lists are sequences of values
- Values in the list are called elements (or items)
- Lists are “mutable”

  -- that means lists can be changed

  -- we will find out about tuples in the next section … tuples are a lot like lists, but they are immutable (i.e. tuples cannot be changed)

- Lists can contain different data types within a single list … e.g., you can have an integer, a float, a string, and even another list in the same list

LIST BASICS

II. HOW LISTS ARE CREATED

- You create lists with a sequence of elements that are enclosed inside of brackets

- list syntax:

  list_variable = [list-value/item1, list-value/item2, …]

LIST BASICS

III. EXAMPLES OF LISTS

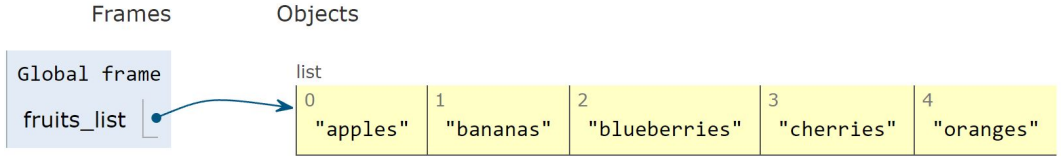
- List of string objects

  fruits_list = [‘apples’, ‘bananas’, ‘blueberries’, ‘cherries’, ‘oranges’]

  print(type(fruits_list))

Visualize Code Execution

```
<class 'list'>
```



LIST BASICS

III. EXAMPLES OF LISTS

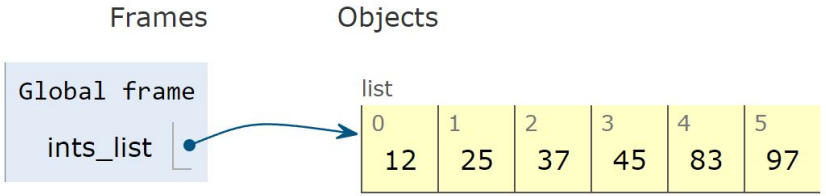
- List of integers

```
ints_list = [ 12, 25, 37, 45, 83, 97 ]

print(type(ints_list))
```

```
<class 'list'>
```

Visualize Code Execution



LIST BASICS

IV. EXAMPLES OF LISTS

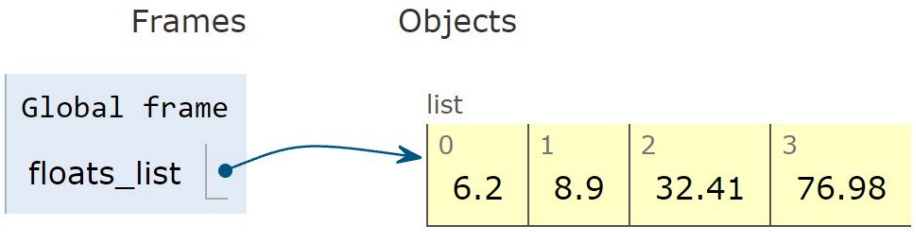
- List of floats

```
floats_list  = [ 6.2, 8.9, 32.41, 76.98 ]

print(type(floats_list))
```

```
<class 'list'>
```

Visualize Code Execution



LIST BASICS

IV. EXAMPLES OF LISTS

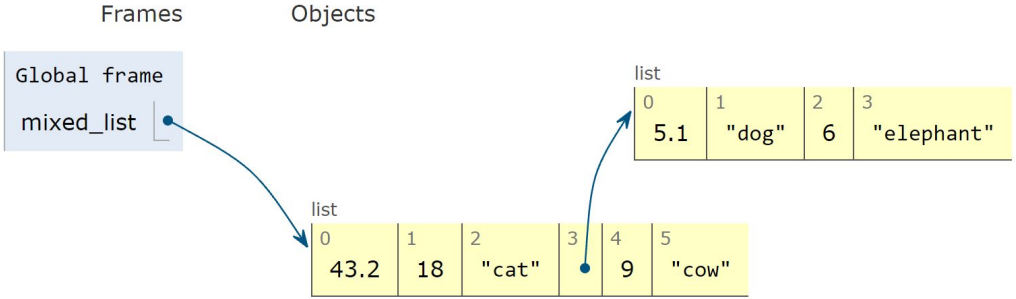
- List with several data types

  mixed_list = [43.2 , 18, “cat”, [5.1, “dog”, 6, “elephant”], 9, “cow”]

  print(type(mixed_list))

```
<class 'list'>
```

Visualize Code Execution



LIST BASICS

V. LIST ITEMS HAVE INDEXES

- Each element in a list has an index
- list indexes start at 0 -- just like strings and other sequences

States_list = [‘Minnesota’, ‘Iowa’, ‘Wisconsin’, ‘Illinois’, ‘Indiana’, ‘Michigan’, ‘Ohio’]

index:	0	1	2	3	4	5	6
states_list:	Minnesota	Iowa	Wisconsin	Illinois	Indiana	Michigam	Ohio

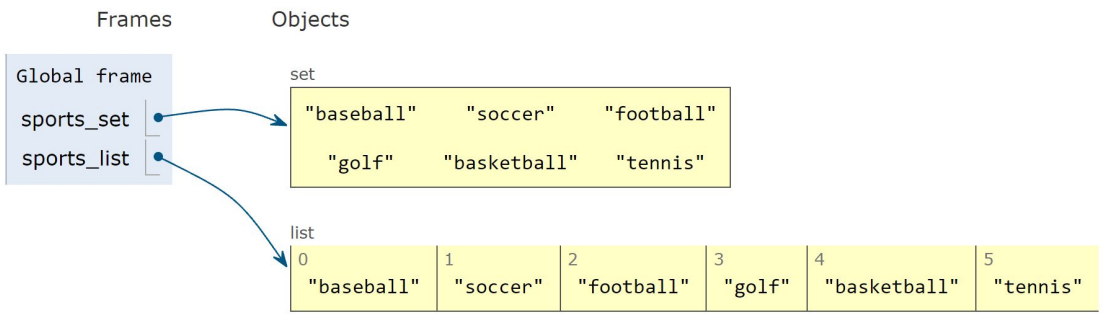
LIST BASICS

VI. OTHER WAYS TO CREATE LISTS

- You can create lists using the list() function
- list() transforms different structures into lists

```
sports_set = {‘football’, ‘basketball’, ‘soccer’, ‘golf’, ‘tennis’, ‘baseball’}
print(type(sports_set))
sports_list = list(sports_set)
print(sports_list)
print(type(sports_list)
```

```
<class 'set'>
['baseball', 'soccer', 'football', 'golf', 'bask
<class 'list'>
```



Visualize Code Execution

LIST BASICS

VII. WHY DO WE USE LISTS?

- Lists can change (lists are mutable)

  -- this is in contrast to tuples

- You can use lists when you have duplicate items

  -- e.g., you have a list of names and two people have the same name

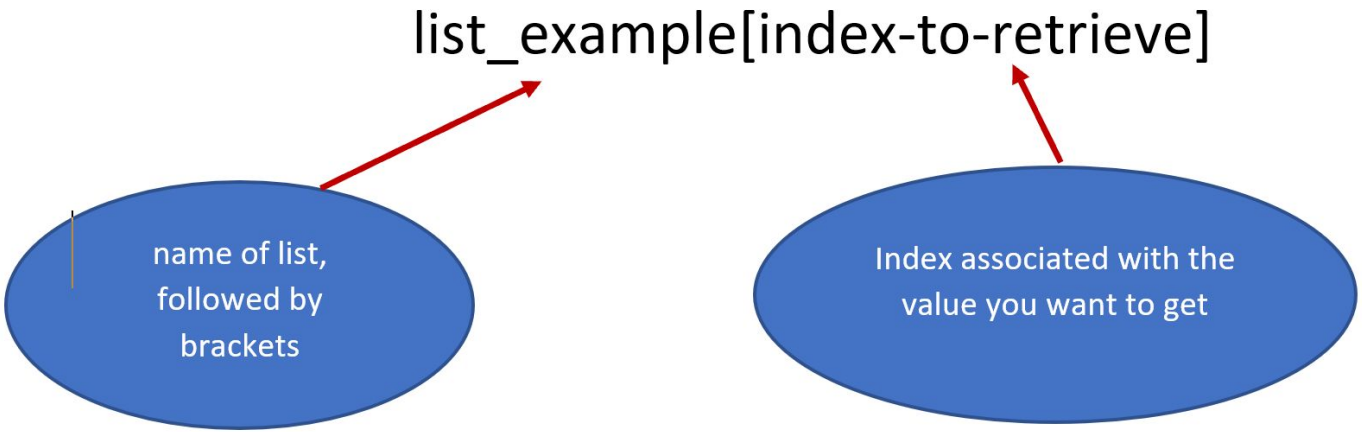
- You can use lists when order matters

  -- lists are ordered

HOW LIST ITEMS ARE RETRIEVED/ACCESSED

I. LIST ITEMS CAN BE ACCESSED USING "BRACKET NOTATION"

- syntax:

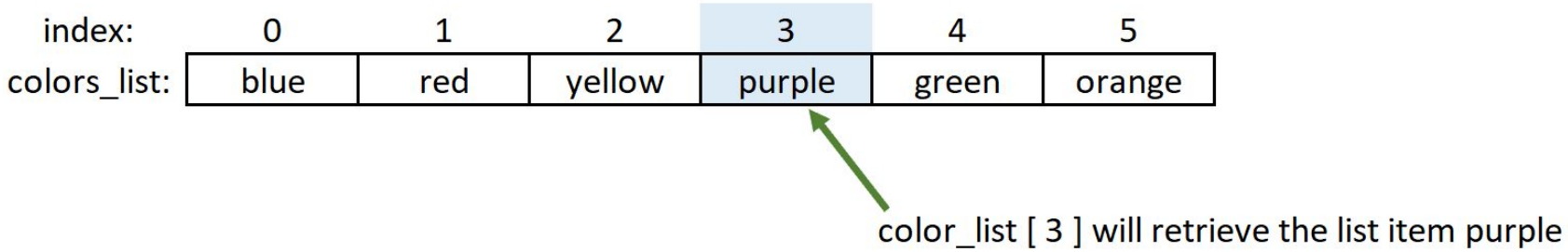


HOW LIST ITEMS ARE RETRIEVED/ACCESSED

II. EXAMPLE: HOW TO RETRIEVE ITEMS FROM A LIST

- colors_list = [“blue”, “red”, “yellow”, “purple”, “green”, “orange”]

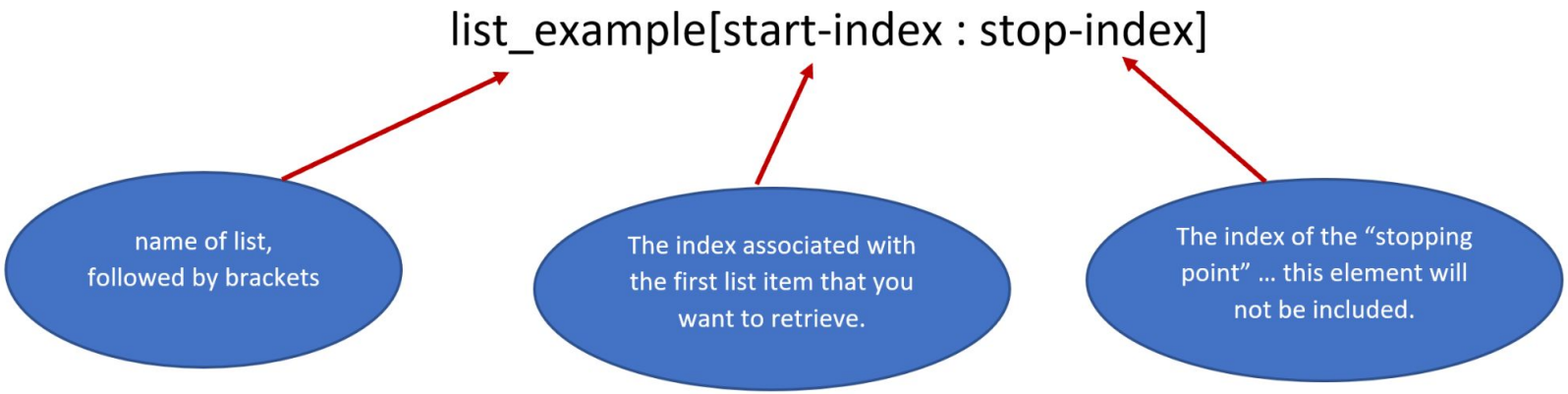
  colors_list [3]



LIST SLICING

I. YOU CAN ALSO RETRIEVE "SLICES" OF LISTS USING BRACKET NOTATION

- this is very similar to slicing strings
- syntax:

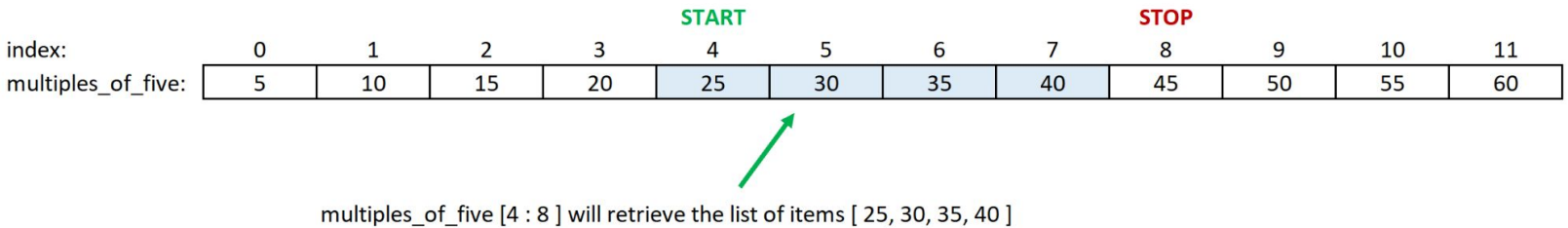


LIST SLICING

II. EXAMPLE: RETRIEVING A SLICE OF A LIST

- multiples_of_ten = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]

  multiples_of_five [4 : 8]

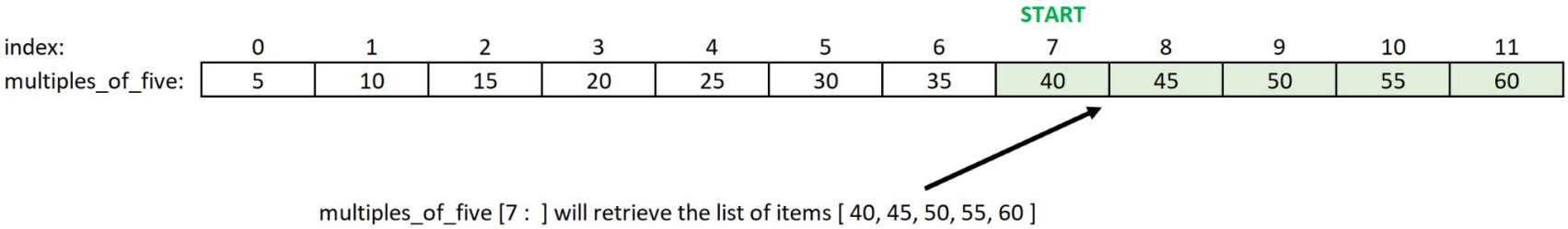


LIST SLICING

III. EXAMPLE: STOP INDEX ISN'T INCLUDED

- this will retrieve the slice that goes from the start index to the end of the list
- multiples_of_ten = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]

  multiples_of_five [7 :]



LIST SLICING

We will learn a lot more about slicing lists when we get further into this unit!!!!

LIST METHODS

- Lists also have “list methods”
- List methods can be used to perform operations on lists

  -- list methods are called using “dot notation”

  -- e.g., alist.append()

method	what it does
`append()`	add an item to end of list
`extend()`	extends a list with a new list of items
`remove()`	removes an item from a list
`sort()`	sorts a list (by value, not by index)
`index()`	returns the index of the first occurrence of the given

* note, this is an *abridged* list of list methods

ADDING ITEMS TO A LIST

- There are two primary ways to add items to a list

  -- append()

  -- extend()

- You can also concatenate strings

ADDING ITEMS TO A LIST

append()

ADDS NEW ELEMENTS TO THE END OF A LIST, ONE AT A TIME

- colors_list = [“blue”, “red”, “yellow”, “purple”, “green”, “orange”]

  colors_list.append(“white”)

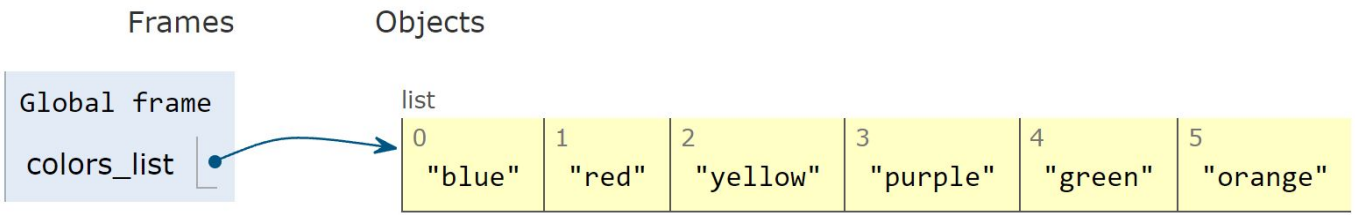
  print(color_list)

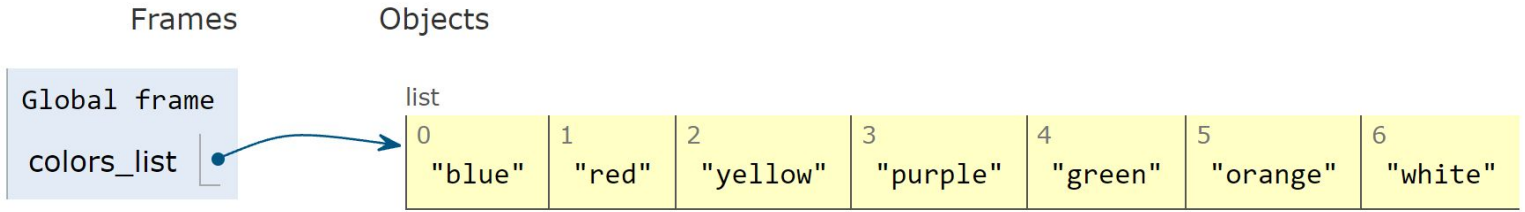
  [“blue”, “red”, “yellow”, “purple”, “green”, “orange”, “white”]

Visualize Code Execution … see next slide

ADDING ITEMS TO A LIST

```
colors_list = ["blue", "red", "yellow", "purple", "green", "orange"]
colors_list.append("white")
print(color_list)
```





ADDING ITEMS TO A LIST

extend()

ADDS MULTIPLE ELEMENTS TO A LIST

- sports_list = [‘football’, ‘basketball’, ‘soccer’, ‘golf’, ‘tennis’, ‘baseball’]

  extra_sports = [‘track & field’, ‘wrestling’, ‘gymnastics’]

  sports_list.extend(extra_sports)

  [‘football’, ‘basketball’, ‘soccer’, ‘golf’, ‘tennis’, ‘baseball’, ‘track & field’, ‘wrestling’, ‘gymnastics’]

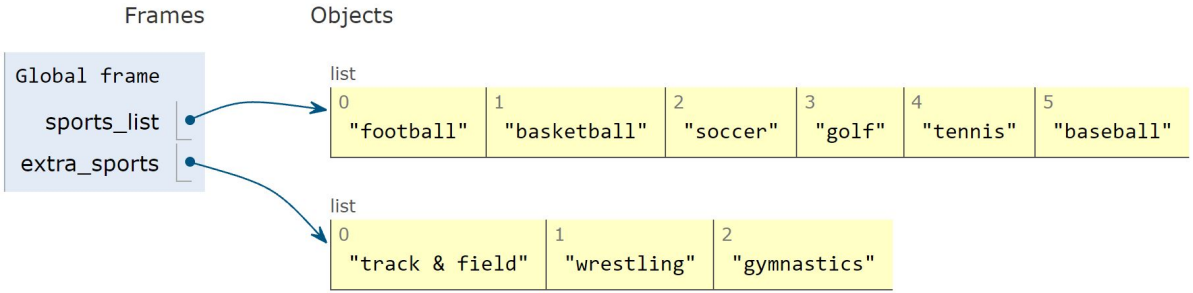
  Visualize Code Execution … see next slide

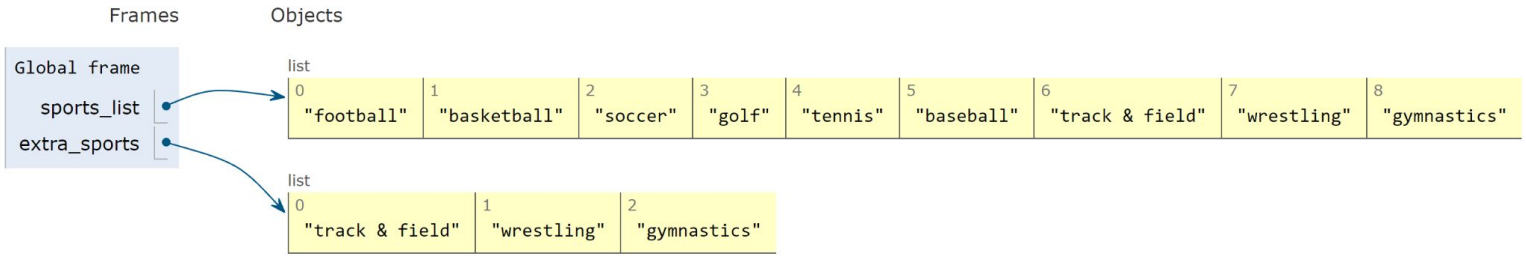
ADDING ITEMS TO A LIST

```
sports_list = ['football', 'basketball', 'soccer', 'golf', 'tennis', 'baseball']

extra_sports = ['track & field', 'wrestling', 'gymnastics']

sports_list.extend(extra_sports)
```





YOU CAN COMBINE EXISTING LISTS TOGETHER WITH THE + OPERATOR

- this is very similar to string concatenation
- fruits_list1 = [‘apples’, ‘oranges’, ‘cherries’]

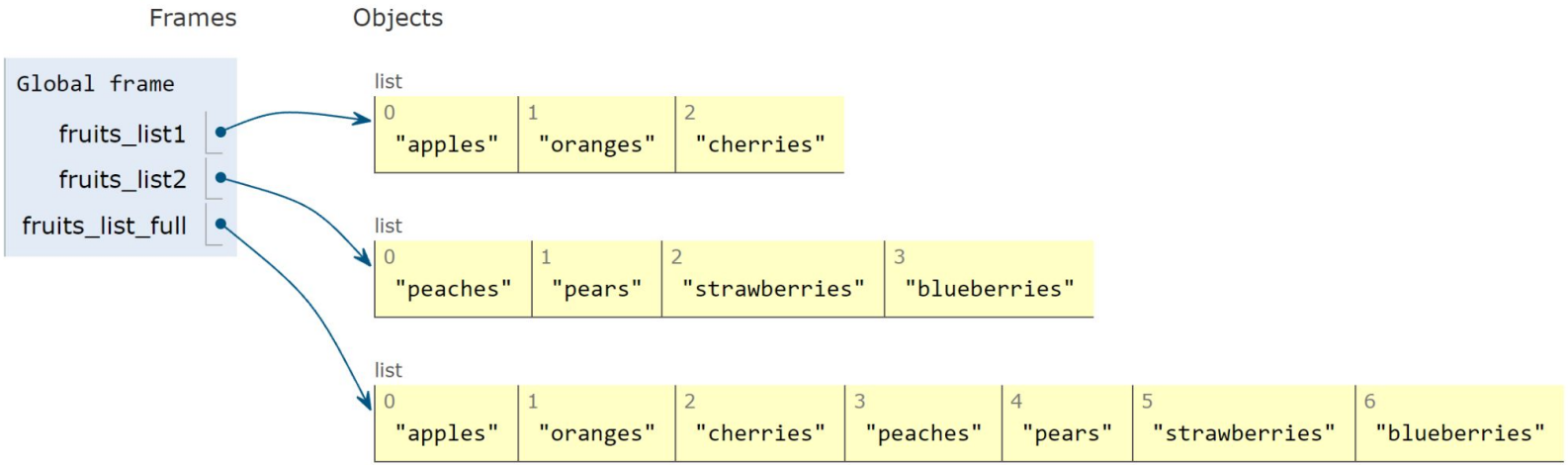
  fruits_list2 = [‘peaches’, ‘pears’, ‘strawberries’, ‘blueberries’]

  fruits_list_full = fruits_list1 + fruits_list2

[‘apples’, ‘oranges’, ‘cherries’, ‘peaches’, ‘pears’, ‘strawberries’, ‘blueberries’]

Visualize Code Execution … see next slide

YOU CAN COMBINE EXISTING LISTS TOGETHER WITH THE + OPERATOR



REMOVING ITEMS FROM A LIST

THERE ARE SEVERAL WAYS TO REMOVE ITEMS FROM A LIST

-- del

-- remove

NOTE:

WHEN ITEMS ARE REMOVED FROM A LIST THE REMAINING ITEMS SHIFT INDEX POSITION.

DELETE AN ITEM FROM A LIST USING THE del STATEMENT

- Indicate which item to remove by referencing its index

- colors_list = [“blue”, “red”, “yellow”, “purple”, “green”, “orange”]

  del colors_list[4]

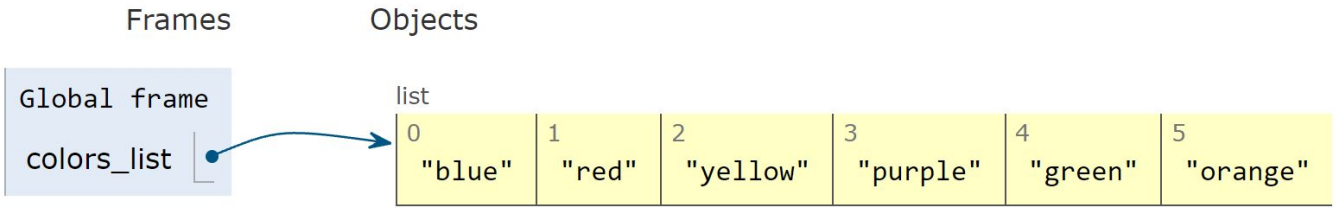
  print(colors_list)

[“blue”, “red”, “yellow”, “purple”, “orange”]

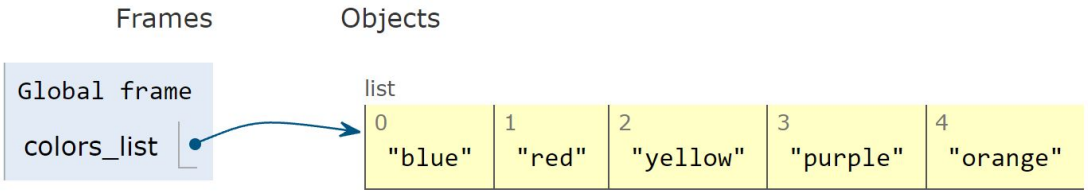
Visualize Code Execution … see next slide

DELETE AN ITEM FROM A LIST USING del

```
colors_list = ["blue", "red", "yellow", "purple", "green", "orange"]
del colors_list[4]
print(colors_list)
```




```
['blue', 'red', 'yellow', 'purple', 'orange']
```



REMOVING AN ITEM FROM A LIST

USING THE .remove() method

- with .remove() the item that is to be removed gets referenced

  -- the .remove() method is useful if you only know the value, not the index

- animals_list = [“dog”, “cat”, “cow”, “elephant”, “giraffe”, “lion”]

  animals_list.remove(“giraffe”)

  print(animals_list)

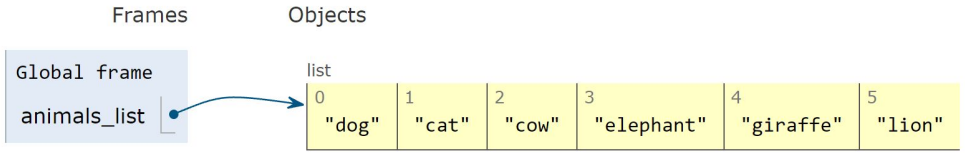
  [“dog”, “cat”, “cow”, “elephant”, “lion”]

  Visualize Code Execution … see next slide

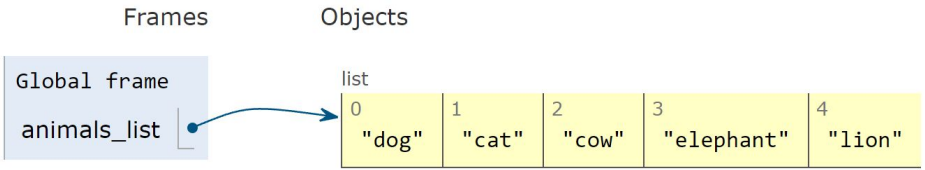
REMOVING AN ITEM FROM A LIST

USING THE .remove() method

```
animals_list = ["dog", "cat", "cow", "elephant", "giraffe", "lion"]
animals_list.remove("giraffe")
print(animals_list)
```




```
['dog', 'cat', 'cow', 'elephant', 'lion']
```



REMOVING ITEMS SHIFTS INDEX POSITION.