Python Lesson 4: Lists and Loops

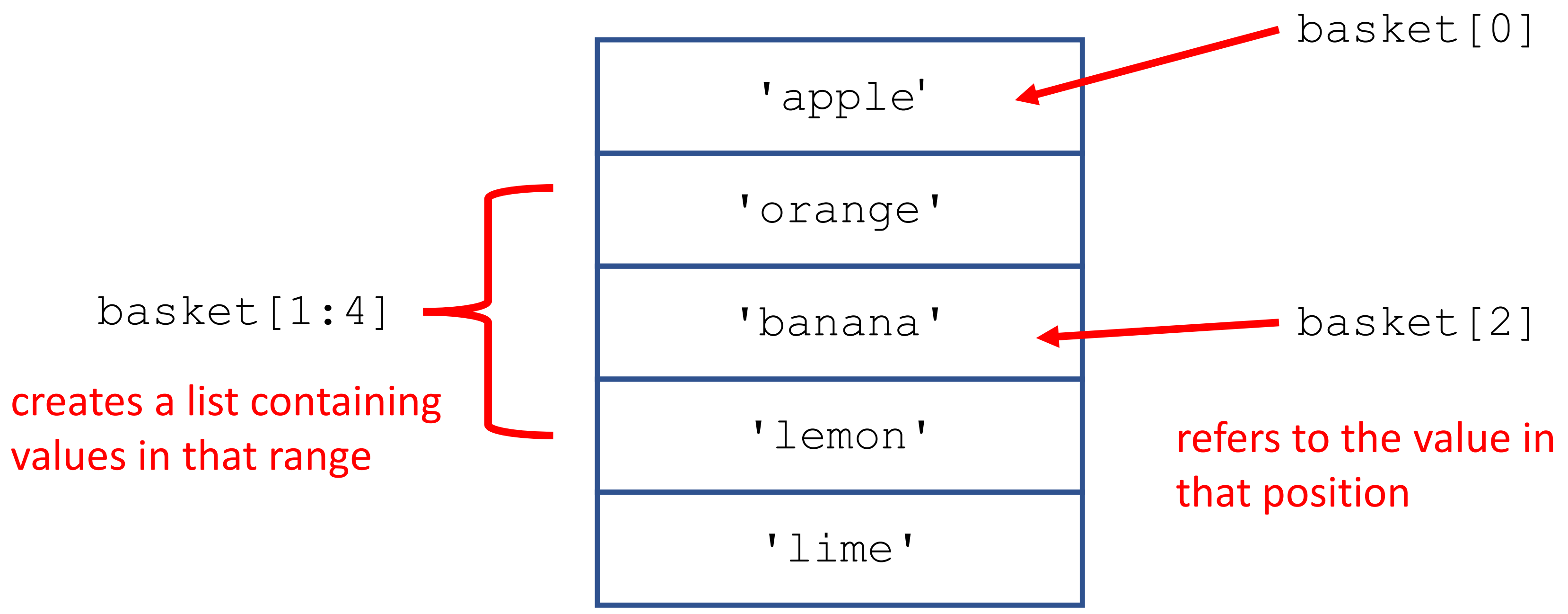
vanderbi.lt/py

Steve Baskauf

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Lists

```
basket = ['apple', 'orange', 'banana', 'lemon', 'lime']
```

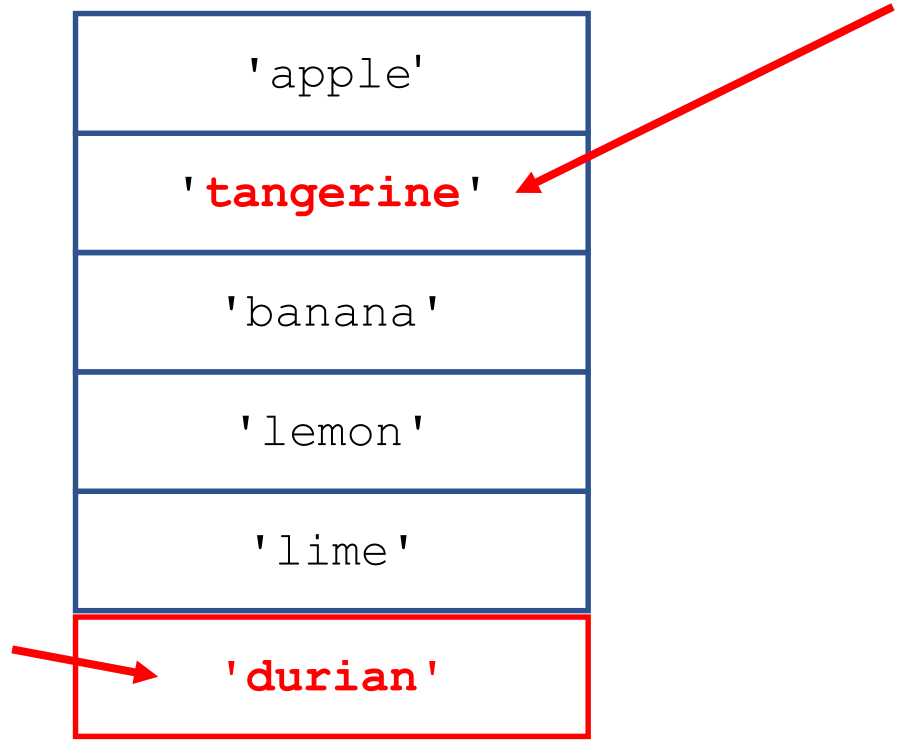

basket[0]
'apple'
'orange'
basket[1:4]
creates a list containing values in that range
'banana'
basket[2]
'lemon'
refers to the value in that position
'lime'

Try this

Changing lists

```
basket = ['apple', 'orange', 'banana', 'lemon', 'lime']
```

```
basket[1] = 'tangerine'
```



```
basket.append('durian')
```

Notice that built-in objects (like lists) can have methods. This `.append()` method does not return a value – it does something.

Lists (more commands)

- Empty list can be created using

```
basket = []
```

- `.remove()` can be used to remove a particular value from the list.
- `del basket[3]` can be used to remove an item by position

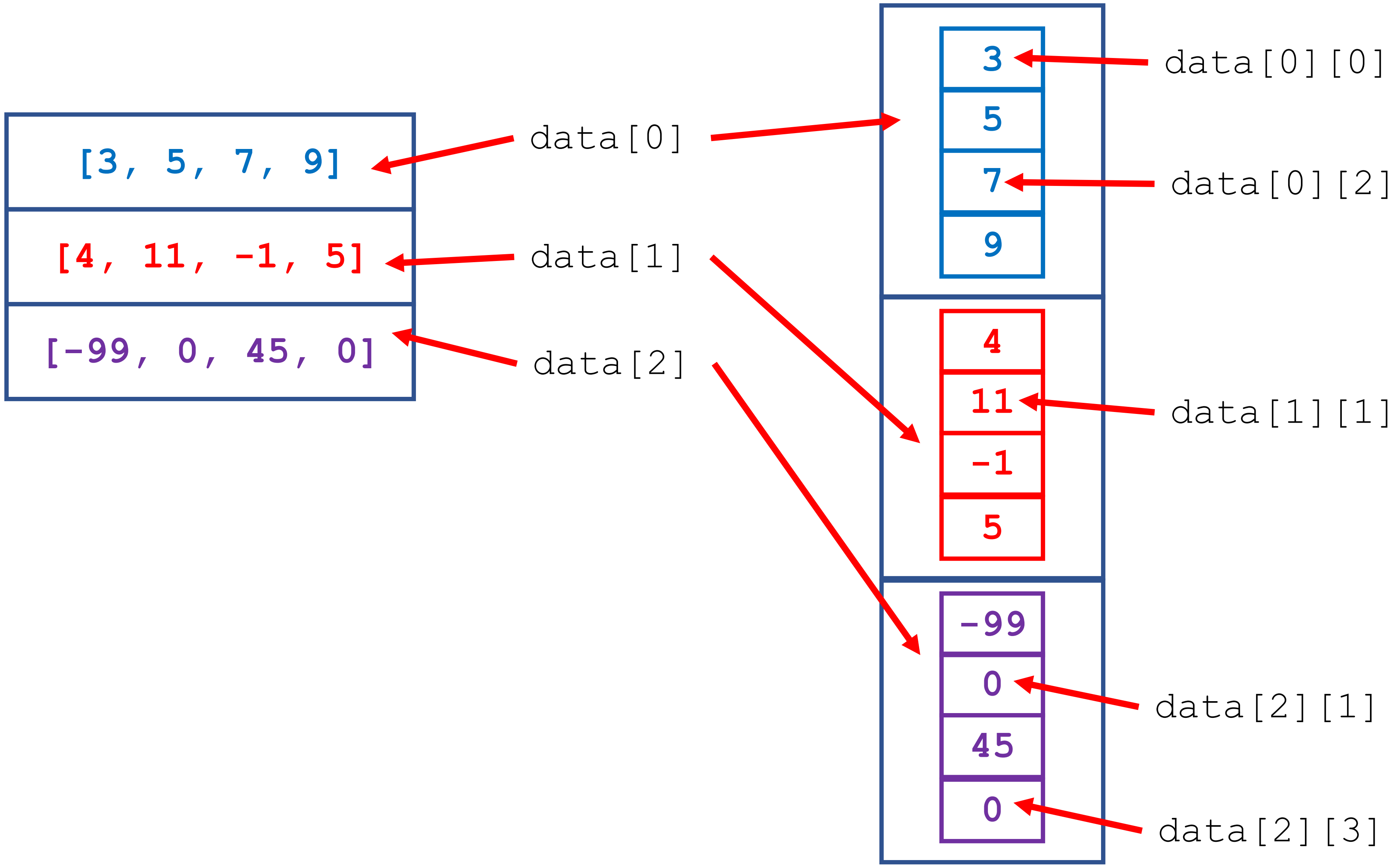
Try this

Important: about copying lists

- As with user-defined objects, lists are complex objects composed of other objects.
- As complex objects, assigning a list to another variable creates a reference from the new variable to the original one. It does NOT make a separate copy.
- To actually make a copy of a list, use the **`deepcopy()`** function from the **`copy`** module.
- Example given on web page.

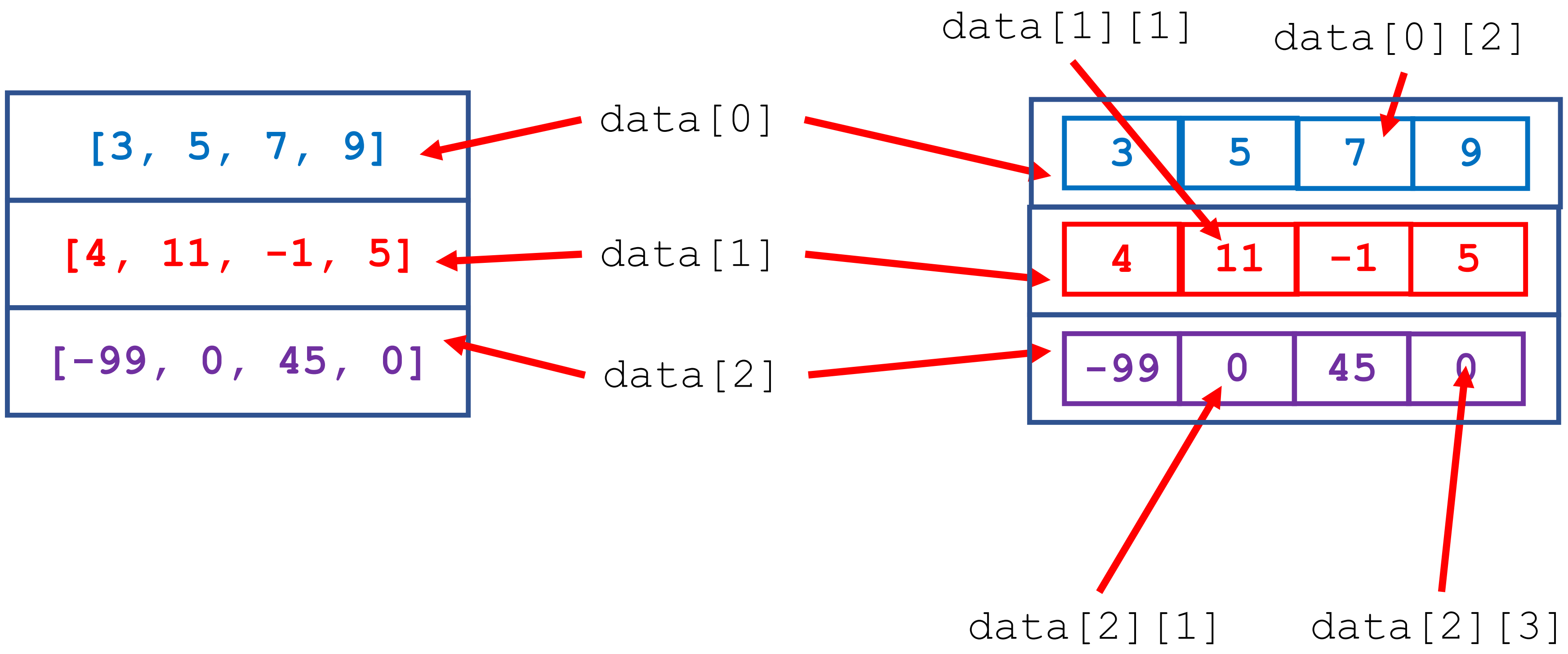
Lists of lists

```
data = [[3, 5, 7, 9], [4, 11, -1, 5], [-99, 0, 45, 0]]
```



Lists of lists

```
data = [[3, 5, 7, 9], [4, 11, -1, 5], [-99, 0, 45, 0]]
```



You can think of this like:

`data[row][column]`

where the indices refer to parts of a table.
A list of lists is similar to an array in other programming languages

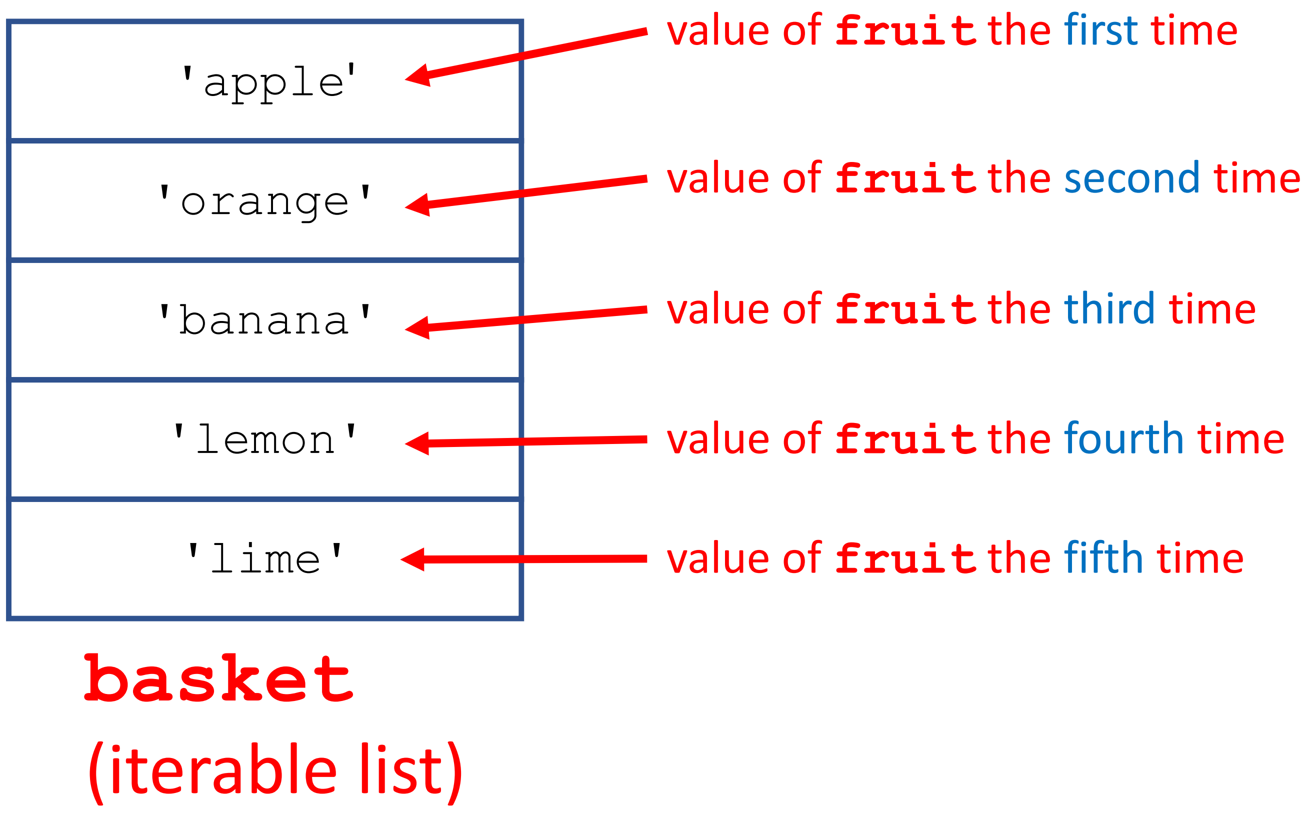
Try this

String manipulations

- Special **escaped** characters: `\n` `\t`
- **Unicode** characters: `\u20ac`
- Substrings: `myWord[3]` `myWord[2:5]`
  - (same issue as lists: one less than final index)
- Methods:
  - `.upper()` **`myWord.upper()`**
  - `.split(',')` **`mySentence.split(',')`**
  - etc.
- Straightforward, try the examples on your own.

Iterating with **`for`**

```
for fruit in basket:
        do this indented code block once for each fruit
then do this code block
```


'apple'
'orange'
'banana'
'lemon'
'lime'
value of fruit the first time
value of fruit the second time
value of fruit the third time
value of fruit the fourth time
value of fruit the fifth time
basket
(iterable list)

Try the example

notice this colon

```
basket = ['apple', 'orange', 'banana', 'lemon', 'lime']
for fruit in basket:
    print('I ate one ' + fruit)
print("I'm full now!")
```

- The indented code block can have more than one line.
- The upcoming code block is signaled by a colon (:) just like **`if`**…**`then`**…**`else`**…

`range()` as an iterable

- The range iterates from the first number to one step less than the second number:
  - `range(1, 11)` iterates from 1 to 10
- A step is optional:
  - `range(2, 10, 2)` iterates by twos from 2 to 8
- The step can be negative:
  - `range(10, 0, -1)` iterates from 10 to 1

Using the value of the range

```
for number in range(1, 11):
    theSquare = number**2
    theArea = theSquare * 3.14159
    print(number, '\t', theArea)
print("Those are the areas all the circles!")
```

- The value of the iterated variable can be used anywhere in the indented code block.
- It's very common to use the length of a list as the end of a range (see last example).
  - This iterates through the whole list because counting is zero-based.

Try this

About homework

- It's highly advisable to try to work through Homework 2.
- We now have the tools available to actually solve a real problem.
- If you can't figure out how to do it, carefully examine each part (A, B, C) to understand how it works.
- Bring questions next week if you don't understand.