Structures: Lists: Complex

CPSC 231: Introduction to Computer Science for Computer Science Majors I
Spring 2021

Jonathan Hudson, Ph.D.
Instructor
Department of Computer Science
University of Calgary

Friday, 30 April 2021

Copyright © 2021

UNIVERSITY OF CALGARY

Slicing

UNIVERSITY OF
CALGARY

Slicing a List

- You can produce copies and sub-lists of a list using the range of indices (:). The following produces a copy of *list* from *a* to *b-1:*

a is the starting index of the slice. The default is 0.

```
list[a:b]
```

b is the ending index of the slice. The default is *len(list)*.
b itself is excluded from the slice.

names[start:end] → to produce a sub-list

names [0]	Marc
names [1]	Ken
names [2]	Jim
names [3]	Tony

- names[:] returns a copy of names
- names[0:2] returns the first two elements in names
- names[-2:] returns the last two elements in names

UNIVERSITY OF CALGARY

Slicing a List

- You can produce a sub-list of a list that consists of certain elements of a list using *:step* in the range of indices

a and *b* are defined in previous slide.

```
list[a:b:step]
```

step is the amount by which a increments. The default is 1. step be positive (increment) or negative (decrement).

names[start:end:**step**]
→ to produce a sub-list

names [0] | Marc
names [1] | Ken
names [2] | Jim
names [3] | Tony

- `names[0:len(names):1]` returns a copy of list
- `names[::]` returns a copy of list
- `names[::-1]` returns a reversed list
- `names[-2::]` returns last two elements
- `names[::2]` returns a list with every other element in names is skipped

UNIVERSITY OF CALGARY

Copy List

Same List

- A list variable is a reference to the list.

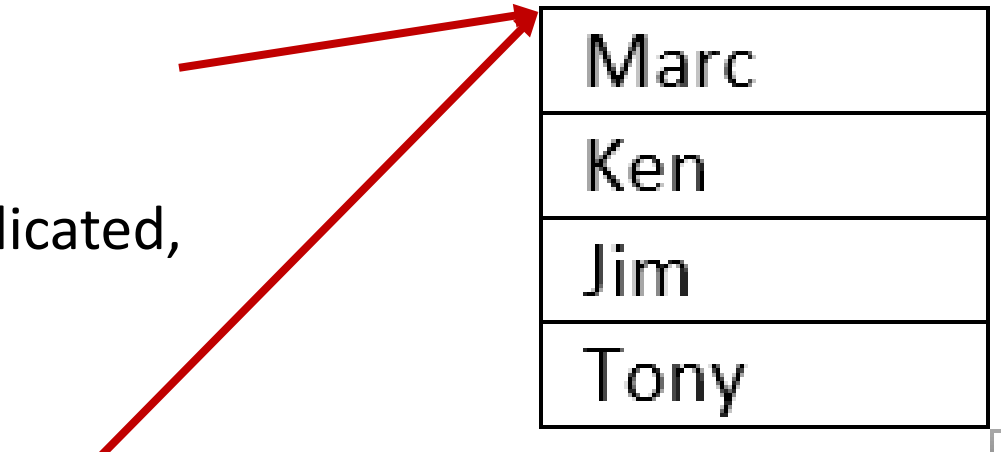 names<address of the first byte of the list in memory>

- When duplicating a list variable, the address is duplicated, not the actual list.

```
>new_names = names
```

If you change *names* you change *new_names*.
Also true the other way.

```
>new_names[0] = "Jonathan"
```

```
>print(names[0]) → 'Jonathan'
```

| Marc |
|------|
| Ken |
| Jim |
| Tony |

UNIVERSITY OF
CALGARY

Passing List to Functions

- When passing mutable types, such as lists, to functions, remember that any changes to the list, will be reflected in the original list in the caller's scope.

```
def func2(list2):
    ...

def func1(list):
    list2 = list
    func2 (list2)


myList = […]
func1(myList)
```

Memory address is passed

memory

Main body | myList | values

func1() | list

func2() | list2

UNIVERSITY OF
CALGARY
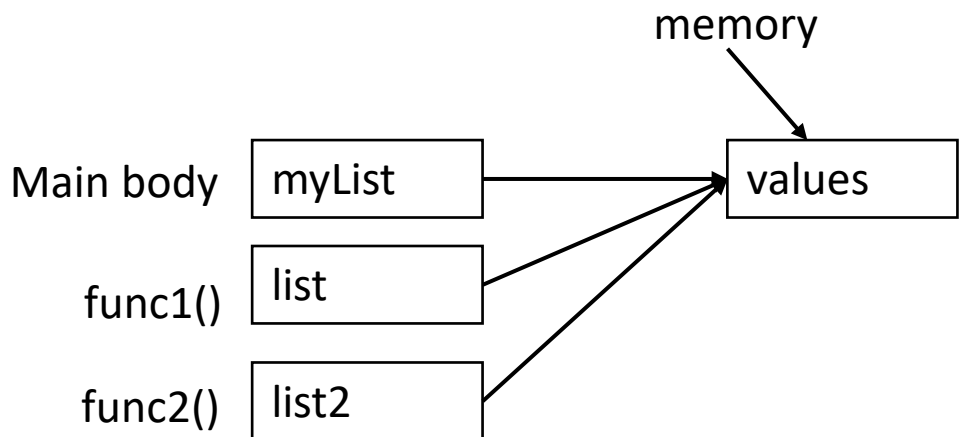
Duplicate a List

- Many ways to create a copy of a list (also known as **shallow-copy**):
  - Using **slice**:
    ```
    new_names = names[:]
    ```
  - Using the **repetition operator**:
    ```
    new_names = names*1
    ```
  - Using **extend()**:
    ```
    new_names = []
    new_names.extend(names)
    ```
  - Using a **loop** to duplicate the list element by element:
    ```
    new_names = []
    for i in range (0, len(names), 1):
        new_names.append(names[i])
    ```

UNIVERSITY OF CALGARY

Operations

UNIVERSITY OF
CALGARY

List Operations and Methods

Operation	Example	Description
Indexing	`names[0]`	Access a list element
Membership	`if 'Alice' in names` `if 'Alice' not in names`	Query whether or not an item is in the list
Length	`len(names)`	Get the number of items in a list
Append	`names.append('Alice')`	Add an item to the end of the list
Insert	`names.insert(0, 'Alice')`	Insert an item at certain position
Sort	`names.sort()`	Sort the list
Reverse	`names.reverse()`	Reverse the items in the list
Count	`names.count('Alice')`	Count the number of occurence of an item

UNIVERSITY OF CALGARY

Search/Remove List

UNIVERSITY OF CALGARY

Searching For Elements

- Use **in** to check if an item is present in a list

  ```
  data = [1,2,3,4,5]
  ```
  `2 in data` evaluates to `True`

  `8 in data` evaluates to `False`

- Use `index` to determine where it is in the list

  ```
  data = [11,12,13,14]
  ```
  `data.index(12)` evaluates to `1`

  `data.index(8)` results in a ValueError

UNIVERSITY OF
CALGARY

Removing Elements

- How can we remove an item from a list?
  - Use the **remove** method
    - **Removes the first occurrence of the item**
    - Subsequent identical items remain in the list
    - Item must exist or a ValueError will occur

```
x = [1,2,1,3,4,2,1]
x.remove(1)
print(x)
```

UNIVERSITY OF CALGARY

Removing Elements

- What if we want to remove all occurrences of an item from a list?

- Use a while loop:

```
while x in myList:
    myList.remove(x)
```

UNIVERSITY OF CALGARY

Removing Elements

- What if we know the index of the item we want to remove?
  - Use *pop(index)*
  - With no parameters: Removes last item
  - With one parameter: Removes item at the index specified
  - Returns the item that is removed

```
myList = [1,2,3,4]

myList.pop()
print(myList)

myList.pop(0)
print(myList)

myList.pop(myList.index(2))
print(myList)
```

[1, 2, 3]

[2, 3]

[3]

UNIVERSITY OF
CALGARY

Sorting a List

Sorting

- Sorting is the process of ordering elements of a list in ascending or descending order.

`[4, 2, 1, 3, 0]` ⟶ **Unordered** list

`[0, 1, 2, 3, 4]` ⟶ Ordered list in **ascending order**

`[4, 3, 2, 1, 0]` ⟶ Ordered list in **descending order**

- How do we sort the list?

UNIVERSITY OF
CALGARY

Sorting

- Sorting is an important task
  - Needed when working with large data sets
  - Frequently occurs as part of other algorithms

- Sorting has been studied extensively
  - Many algorithms, some of which are quite complex

UNIVERSITY OF CALGARY

Sorting - Bubble Sort

General idea (ascending order)

- go through list from beginning to end
    - compare adjacent elements
    - swap if previous element is larger than current element

- repeat until no swaps are performed

https://www.youtube.com/watch?v=nmhjrI-aW5o

- You can download a solution: *1_Bubble.py*

UNIVERSITY OF
CALGARY

Sorting - Selection Sort

General idea (ascending order): The list is initially considered entirely unordered.

- Select the smallest element in the unordered portion of list

- Remove the element from unordered portion of the list and place it at the end of the ordered portion of the list.

- Repeat until no elements remain in the unordered portion of the list.

https://www.youtube.com/watch?v=xWBP4lzkoyM

Lets implement this!

You can download another solution: *2_Selection.py*

UNIVERSITY OF
CALGARY

Sorting in Python

- Python makes sorting a list easy
  - Use the sorted function
    - Takes one parameter which is an unsorted list
    - Returns a new list sorted into increasing order
  - Use the *sort(order)* method
    - Order is a Boolean parameter. Default is True for ascending order. False sorts in descending order.
    - Invoked on a list using dot notation
    - Modifies the list

```
list=[0,1,6,2,10]
list.sort()
print(list)            ⟶      [0, 1, 2, 6, 10]
```

UNIVERSITY OF
CALGARY

List Example

UNIVERSITY OF CALGARY

Practice Example

- Compute the median of a list of values entered by the user
  - User will enter an unknown number of values
  - A blank line will be used to indicate that no additional values will be entered
  - If the list has an odd number of elements
    - Median is the middle value
  - If the list has an even number of elements
    - Median is average of the two middle values

Practice Example Design

- read values from user and store in a list (using append)

- sort list (put numbers in ascending order)

- if list length is odd, display middle value (index = `len(list)/2`)

- if list length is even, display the average of two middle values (index len(list)/2 and len(list)/2 – 1)

Lets code this!

UNIVERSITY OF CALGARY

Tracing

UNIVERSITY OF CALGARY

Trace The Code 1:

```python
def f1(list1) :
    list2 = list1
    for index in range(len(list1)):
        list2[index] = list1[index]+1
    print(list1)
    print(list2)


f1([1,2,3])
```

➡️ [2, 3, 4]
[2, 3, 4]

UNIVERSITY OF CALGARY

Trace The Code 2:

```python
def f2(list1) :
    list2 = list1[:]
    for index in range(len(list1)):
        list2[index] = list1[index]+1
    print(list1)
    print(list2)


f2([1,2,3])
```


[1, 2, 3]
[2, 3, 4]

UNIVERSITY OF CALGARY

Trace The Code 3:

```python
def f3(list1) :
    list2 = list1*2
    for index in range(len(list2)):
        list2[index] += 1
    print(list1)
    print(list2)


f3([1,2,3])
```
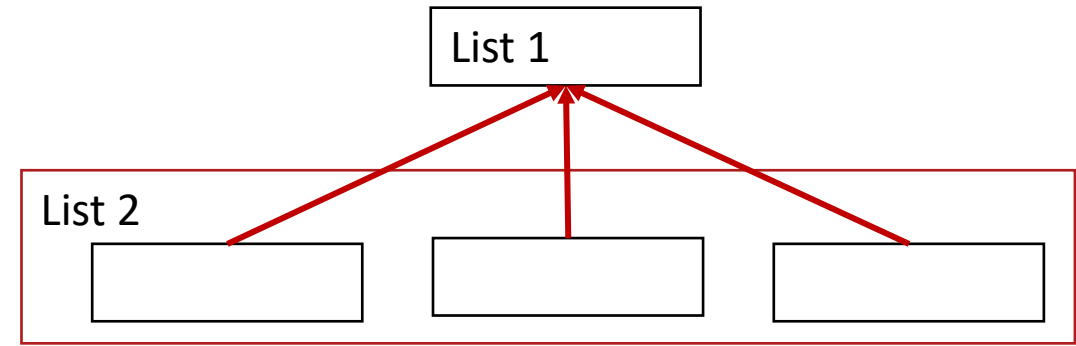


```
[1, 2, 3]
[2, 3, 4, 2, 3, 4]
```

UNIVERSITY OF
CALGARY

Trace The Code 4:

```
def f4(list1) :
    list2 = [list1]*3
    for index in range(len(list2)):
        innerList = list2[index]
        for innerIndex in range(len(innerList)) :
            innerList[innerIndex] += 1
    print(list1)
    print(list2)
```

List 1

List 2

```
f4([1,2,3])
```

➡️ [4, 5, 6]
[[4, 5, 6], [4, 5, 6],[4, 5, 6]]

UNIVERSITY OF CALGARY

Trace The Code 5:

```python
def f5(list1) :
    list2 = [list1]*2
    for index in range(len(list2)):
        innerList = list2[index]
        innerList = innerList[:]
        for innerIndex in range(len(innerList)) :
            innerList[innerIndex] += 1
        list2[index] = innerList
    print(list1)
    print(list2)


f5([1,2,3])
```

```
[1, 2, 3]
[[2, 3, 4], [2, 3, 4]]
```

UNIVERSITY OF CALGARY

Onward to ... sets and tuples.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~hudsonj/

UNIVERSITY OF
CALGARY