Lecture 10: Lists (cont'd)

CS 51P

October 5, 2022

Tom Yeh
he/him/his

Class News

- Checkpoint 1 on Monday 10/10
  - Review session by TAs

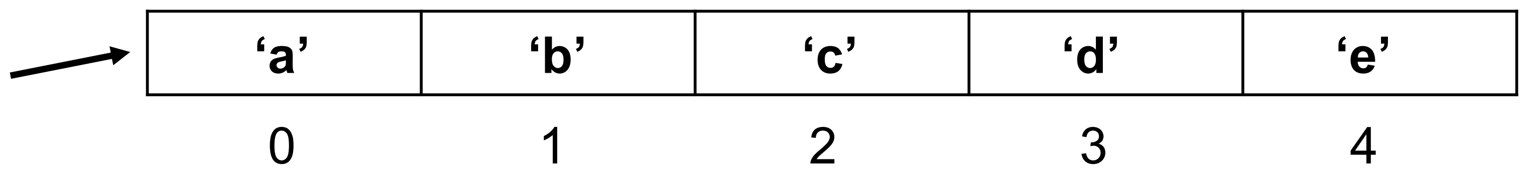
- No assignment this week ☺
  - This week’s lab is an ethics debate

Learning Goals

- Practice coding with lists
- Learn about tuples

Lists

- a list is an ordered set of elements:
- a_list



- many ways to create a list including:

```
a_list = [3, 6, 2, 1]
b_list = []
c_list = "a b c d".split()
d_list = open("temp.txt","r").readlines()
```

- a list is a sequence, so can index into, loop over, check for membership, slice, etc
- operators: + and *
- lists are mutable

List Operations

adding to a list
(updates original list)

- a_list.extend(*list*)
- a_list.append(*elem*)
  - Different than extend – e.g. [5, 1]
- a_list.insert(*index*, *elem*)

other

- min(a_list), max(a_list), len(a_list)
- *elem* in a_list
  - returns bool
- a_list.index(*elem*)
  - returns index of 1st instance of elem or error
- a_list.insert(*index, elem*)
  - Insert elem at index, shifts down
- a_list.copy()
  - Returns a copy of list
- if a_list:
  - checks is list is empty

List Operations

removing from a list

- del(a_list[*slice*])
- a_list.remove(*elem*)
  - removes 1st instance of elem
  - error if *elem* not in a_list
- a_list.pop()
  - returns (and removes) a_list[-1]
- a_list.pop(*index*)
  - returns (and removes) a_list[index]

modifying a list

- direct assignment
  - a_list[0] = 2

printing a list

```
>>> print(a_list)
[1, 2, 3, 4, 5]
```

+ and * operators

- Works on lists, but creates a new list
  - >>> a_list = [1, 2, 3]
  - >>> new_list = a_list + a_list
  - >>> new_list
  - [1,2,3,1,2,3]

Code Examples

- num_list = [1, 2, 3, 4]
- x = 5
- How do we check to see if num_list is empty?

- How do we check if num_list contains x?

- How do we store the value of the last element in x?

- How do we store the value of the last element in x and remove it from the list?

- How do we add the value in x to num_list?

More Code Examples

- num_list = [1, 2, 3, 4]
- second_list = [5, 1]

- What does num_list.insert(2, 51) do?

- How do we remove the first 1 from the combined list?

- How do we combine the two lists? Two ways.

Even More Code Examples

- num_list = [1, 2, 3, 4]
- second_list = [5, 1]

- third_list = num_list + second_list
- Using the + and * operator works like extend, but it creates a new list. Original lists are unchanged. Need to assign it to a variable.

List.copy

- list.copy() – returns a copy of the list

- >>> sports = [’tennis’, ‘basketball‘, ‘swimming’, ‘soccer’]
- >>> my_sports = sports.copy()
- >>> my_sports.insert(‘running’)
- >>> my_sports
- ???
- >>> sports
- ???

Assigning a list to another

- >>> sports = [’tennis’, ‘basketball‘, ‘swimming’, ‘soccer’]
- >>> my_sports = sports
- >>> my_sports.insert(‘running’)
- >>> my_sports
- ???
- >>> sports
- ???

min(list) and max(list)

- Returns max value in the list
- >>> numbers = [1, 2, 4, 8]
- >>> numbers.max()
- 8
- >>> numbers.min()
- 1

Looping Through List Elements

- food_list = [‘bacon’, ‘bread’, ‘egg’]

- For loop using range:
  - for i in range(len(menu_list)):
    - elem = menu_list[i[
    - print(elem)

- For-each loop
  - for elem in menu_list:
    - print(elem)

- Both loops iterate through all elements of the list
  - variable elem is set to each element in the list in order

Exercise

- Define a function digits that takes one parameter num (an positive int) and returns a list of the digits of num

Example

- Define a function word_list that takes a filename as an argument and returns a list of all the words in that file.

Example – why do we use extend?

- Define a function word_list that takes a filename as an argument and returns a list of all the words in that file.

```
• def word_list(filename):
•   file = open(filename, "r")
•   words = []
•   for line in file:
•     words_in_line = line.split()
•     words.extend(words_in_line)
•   file.close()
•   return words
```

Exercise

- Define a function count_words that takes a filename as input and returns the total number of unique words in that file

Example

- write a function `odds` that takes a list of ints and returns a list of the odd elements

Example

- write a function `odds` that takes a list of ints and returns a list of the odd elements

- `def odds(int_lst):`
  - `odd_lst = []`
  - `for num in int_lst:`
  - `if num % 2 == 1:`
    - `odd_lst.append(num)`
  - `return odd_lst`

Exercises

- write a function `double` that takes a list of ints and returns a list with every number doubled
- write a function `max` that takes a list of ints and returns the largest value

Questions?

Tuple – another built-in data type

- a tuple is a way to keep track of an *ordered collection* of items
  - Similar to a list, but **immutable** (can not be changed in place)
  - **Ordered**: can refer to elements by their position (start with 0)
  - **Collection**: tuple can contain multiple items

```
num_tuple = (1, 2, 3)
```

- Often used to track data that are related:
  - Coordinates for a point: (x, y)
  - RGB values for a color: (red, green, blue)

- Can be used to return multiple values from a function

Creating Tuples

- Creating tuples
  - Tuples start/end with parenthesis with elements separated by commas.

```
random_tuple = (3, 6, 2, 1)
point = (5.1, 6.2)
addr = (‘333 N College Way’, ‘Claremont’, ‘CA 91711’)
empty_tuple = ()
```

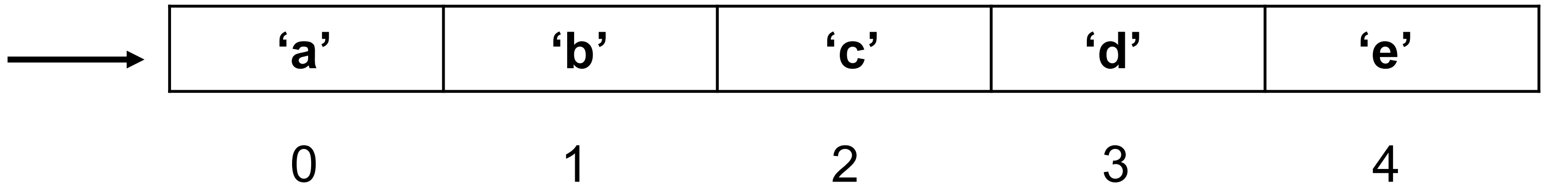
- Tuple with 1 element **is** the same as the element
  - >>> tuple_one = (51)
  - >>> one = 51
  - >>> tuple_one == one
  - True

Accessing Elements of a Tuple

- Consider this tuple:

```
a_tuple = ('a', 'b', 'c', 'd', 'e')
```

- Access elements of tuple just like the list
  - Index starts from 0
- a_tuple



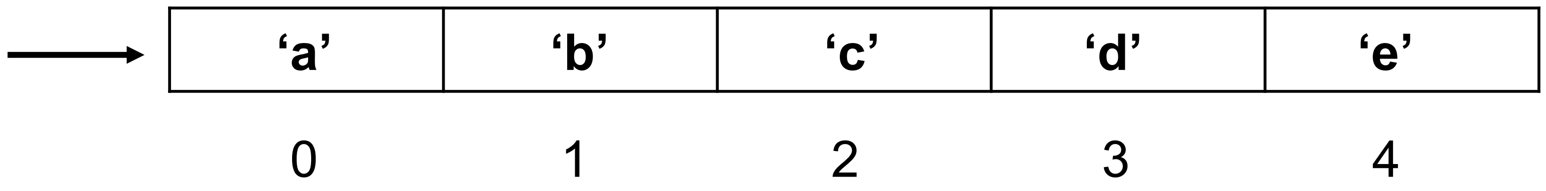
- Accessing individual elements:
  - a_tuple[0] is ‘a’
  - a_tuple[3] is ‘d’

Accessing Elements of a Tuple

- Consider this tuple: `a_tuple = ('a', 'b', 'c', 'd', 'e')`

- Access elements of tuple just like the list
  - Index starts from 0

- a_tuple



- Can **not** assign to individual elements:
  - Tuples are immutable
  - a_tuple[0] = 'x'

  - TypeError: 'tuple' object does not support item assignment

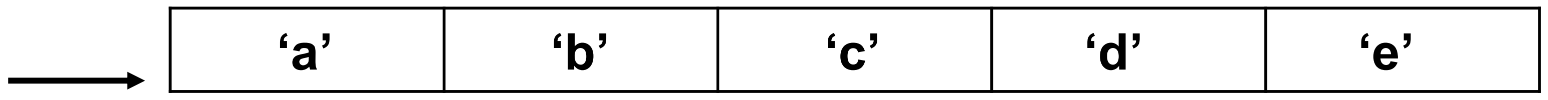
Accessing Elements of a Tuple

- Consider this tuple: :

```
a_tuple = ('a', 'b', 'c', 'd', 'e')
```

- Access elements of tuple just like the list
  - Index starts from 0

- a_tuple



- Can **not** assign to individual elements:
  - Tuples are immutable
  - No append/pop functions

- To change a tuple, we need to create new tuple and overwrite variable
  - a_tuple = a_tuple[0:2]

Similar to lists

- Same for
  - Indexing
  - slicing
  - checking for empty tuple
  - checking if tuple contains an element
  - same ways with for loop to iterate through tuples

- Few functions
  - Min, max, sum

Assignment with tuples

- Can use tuples to assign multiple variables at the same time
  - Number of variables on left hand side needs to be the same as the right hand side

  - >>> (x, y) = (5, 1)
  - >>> x
  - 5
  - >>> y
  - 1

Tuples and List

- Can create tuple from list
- >>> num_tuple = (1, 2, 3, 4, 5)
- >>> num_list = list(num_tuple)
- >>> num_list
- [1, 2, 3, 4, 5]

- Can create list from tuple
- >>> a_list = [’Red’ , ‘Green’ , ‘Blue’]
- >>> a_tuple = tuple(a_list)
- >>> a_tuple
- (‘Red’ , ‘Green’ , ‘Blue’)

Why Tuples?

- More restrictive because it is immutable
- Tuples are more memory efficient than lists
- Execution speed of using tuples is faster than using lists

Learning Goals

- Practice coding with lists
- Learn about tuples