CSE 8A
Week 3 Discussion

PA3 - Lists And Strings

Sean Ran

Reminders

- PA1 Redo is due next Tuesday
- PA2 was due last night
  - We'll release feedback on your coding style next week along with PA2 Redo
- Lab3 is tomorrow - make sure to attend
- PA3 release today - start early
  - The deadline is **next Friday** because of the upcoming midterm next Wednesday

List - Create

my_list1 = [1, 2, 3]

my_list2 = [“1”, “2”, “3”]

my_list3 = [“Lisa”, “John”, “Amy”, “Jack”]

my_list4 = []

1. What are the values of **len(my_list1)**, **len(my_list3)** and **len(my_list4)**?
2. What is the difference between my_list1 and my_list2?
3. Consider [1, “Tom”, my_list3], is this a valid list?

List - indexing

List elements all have there unique index in the list, starting from 0 to **(length of the list - 1)**

my_list3 = [“Lisa”, “John”, “Amy”, “Jack”]

0	1	2	3
-4	-3	-2	-1

Given a list, you can get a single element at any index by indexing, e.g.

my_list3[2] gives “Amy”

What about **my_list3[100]**? Array index out of bound error.

What about **my_list3[-3]**? "John"

List - indexing (reversed)

What if we are given my_list3 = [“Lisa”, “John”, “Amy”, “Jack”] and “Amy”, and we want to know the index of “Amy” inside my_list3?

There is a built-in function for list in python:

my_list3.index(“Amy”)

Wait, shouldn’t all function calls be like function_name(argument_list)?

E.g. print, len ...

Detour - another way of calling function

Object calling a function using following format:

calling_object.function_name(argument_list)

The function's behavior depends not only on argument_list but also calling_object.

Example: list1 = [1, 2, 3], list2 = [3, 2, 1, 2]

Then **list1.index(1)** gives 0 but **list2.index(1)** gives 2

What about **list2.index(2)**?

Some other frequently used list operations

Keyword **in** - check whether an element is in the list or not

Function **append** - adding a new element to the end of a list

Function **count** - check the number of times an element occurs in a list

List slicing - get a sub-list of original list based on starting and ending index

Live Demo time!

String

my_string = “CSE 8A”

my_list = [“C”, “S”, “E”, “ “, “8”, “A”]

Do you see any similarities between characters in a string and elements in a list? - Length? Indexing? And more?

What if you were the inventor of Python?

Why not use same format for both list operations and string operations that are in common?

Some frequently used string operations

Spoiler alert: you may need to use a lot of them in PA3!

String indexing - get the character at certain index of the string

Keyword **in** - check whether a character is in the string or not

Function **index** - get the (first) index of certain character in string

Function **count** - check the number of times a character occurs in a string

String slicing - get a substring of original string based on starting and ending index

String comparison - use **==** to check if two strings are the same

Function **strip** - remove all starting and ending spaces of a string

Attend Lab 3 for more details!

PA3

dining_hall_menus.py

count_exact_match.py

validate_email.py

Extra credit problem

dining_hall_menus - what

Given two lists: **ovt_menu** and **pines_menu**, the function returns the difference between the two menus offered that day.

i.e. A list of **unique items** that one menu has but the other menu doesn't. E.g. when

```
ovt_menu=["fried chicken","ramen","sushi"]
```

```
pines_menu=["ramen","sushi","pizza","pizza"]
```

should return `["fried chicken","pizza"]`

dining_hall_menus - How

1. Create a `result_list` and initialize it to empty list
2. Use loop to iterate through every element in one menu
    a. Check whether that element is in the other menu
    b. Check whether that element is in already in `result_list` (we don't want duplicates!)
    c. If both steps a and b give False, add this element to the end of `result_list` (how?)
3. Use another loop to iterate over the other menu -- same idea!
4. Return `result_list` after loops

count_exact_match - What

Given two arbitrary strings, `text` and `query`, return the count of "exact matches" between them.

An "exact match" is here defined as the case where the same

character appears at the same index of both strings. E.g.

index	0	1	2	3	4	5	6	7	8	9	10	11
text	s	e	v	e	r	u	s	s	n	a	p	e
query	a	l	a	n	r	i	c	k	m	a	n	

`count_exact_match("severussnape", "alanrickman")` returns 2

count_exact_match - How

1. Initialize `result` = 0
2. Loop through all indices for the strings (from 0 to what?)
    a. Get characters at that index from both strings
    b. If these two characters are the same, increment `result`, otherwise do nothing
3. Return `result` after the loop

validate_email - What

Given two strings `email_address` and `email_suffix` and returns `True` if the email address is valid and `False` otherwise. Email is considered valid if all three rules are satisfied:

1. Contains exactly one '@'
2. There is no space in the middle, regardless of leading and ending spaces
3. The substring after '@' and before ending spaces should match `email_suffix`

If `email_suffix` is "ucsd.edu", which of following is a valid email?

A. "tom@tom@ucsd.edu" B. " tom@ucsd.edu " C. " to m@ucsd.edu"

D. "tom$ucsd.edu" E. "tom@gmail.com "

validate_email - How

1. Remove all starting and ending spaces to get a new string `striped_email`
2. Make sure there is only one '@' in `striped_email`
3. Make sure `striped_email` doesn't contain any space character
4. Get the index of ‘@’ and extract the substring of `striped_email` after ‘@’. Then compare the substring with `email_suffix`

Important

Do not use third party libraries for PA3, what you already learned in lectures and labs will be sufficient to finish the PAs.

Q&A