APS106

UNIVERSITY OF TORONTO

Tutorial 7 – Week 8

We'll be starting at the 10 minute mark

if nothing else, write #cleancode

Agenda

- Lecture review
  - List fundamentals
  - List methods
  - List operators
  - List iteration
- Practice questions

Learning objectives

After this tutorial, learners should be able to:

- recognize / describe / create objects of type list
- access and modify individual elements of a list using subscription and slicing
- insert and remove elements from the list using appropriate list methods/operators
- test whether a value/object belongs to a list using the membership operator
- sort a list using the sort method
- Iterate over lists
- understand the concept of aliasing and use it effectively

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Python list fundamentals

if nothing else, write #cleancode

Python lists

- Lists are ordered collections of values
- Lists elements may be of any type, including of type **`list`** !

  - `["apple", "banana", "potato"]`
  - `[17, -4, 23]`
  - `["Hello world", -17.0, 23, "March-2022"]`
  - `[["APS106", "MY", 150], ["APS106", "BA", 1190]]`

a "nested" list

lists that contain elements which are lists are called **nested lists**.

Creating lists

Using the literal notation [ ]

```
>>> my_list = []
>>> my_list
    []
>>> my_list = ["a", "b", "c"]
>>> my_list
    ['a', 'b', 'c']
```

Using built-in function list() #less common

```
>>> my_list = list()
>>> my_list
    []
>>> my_list = list('abc')
>>> my_list
    ['a', 'b', 'c']
```

Creating lists using helper variables

```
>>> a = 1
>>> b = 2
>>> my_list = [a, b]
>>> my_list
    [1, 2]
>>> s1_grades = [72, 100, 52]
>>> s2_grades = [89, 77, 81]
>>> s3_grades = [44, 82, 90]
>>> grades = [s1_grades, s2_grades, s3_grades]
>>> grades
    [[72, 100, 52], [89, 77, 81], [44, 82, 90]]
```

Create a **nested list** whose elements are `s1_grades`, `s2_grades`, and `s3_grades`

Built-in Functions

Several of Python's built-in functions can be applied to lists, including:

- `len(list)` : return the number of elements in list (i.e. the length)
- `min(list)` : return the value of the smallest element in list.
- `max(list)` : return the value of the largest element in list.
- `sum(list)` : return the sum of elements of list (list items must be numeric).

List methods

- Recall that a method is a function associated with an object
- You can find out the methods associated with type `list` by:
  - typing `dir(list)` in the Python shell
  - searching the online Python documentation

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__',
'__delitem__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__gt__',
'__hash__', '__iadd__', '__imul__', '__init__',
'__iter__', '__le__', '__len__', '__lt__', '__mul__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__reversed__', '__rmul__', '__setattr__',
'__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'count', 'extend',
'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

Adding elements to a list

Recall that strings, integers, and booleans are immutable !!

Python lists are ***mutable***, i.e., their contents can be changed

Lists can be modified in several ways, e.g., using the concatenation operator + or using list methods such as `append()` and `extend()`:

`append()`: Add an item to the end of a list

```
>>> fruits = ["apple", "banana", "pear"]
>>> fruits.append("blueberry")
>>> fruits
    ['apple', 'banana', 'pear', 'blueberry']
```

`extend()`: Add one or more items to the end of a list

```
>>> veggies = ["asparagus", "broccoli"]
>>> fruits.extend(veggies)
>>> fruits
    ['apple', 'banana', 'pear', 'blueberry', 'asparagus','broccoli']
```

Removing elements from a list

List method **`remove()`** removes the first occurrence of an item from a list

```
>>> fruits.remove("pear")
>>> fruits
    ['apple', 'banana', 'blueberry', 'asparagus’, 'broccoli']
```

List Operators[1]

subscription selects an item of a sequence

Accessing list elements using **subscription**, i.e., the **indexing operator**:

```
>>> my_list = ["apple", "banana", "potato"]
>>> my_list[1]
    'banana'
>>> grades
    [[72, 100, 52], [89, 77, 81], [44, 82, 90]]
>>> grades[0]
    [72, 100, 52]
>>> grades
    [[72, 100, 52], [89, 77, 81], [44, 82, 90]]

>>> grades[0][0] + grades[0][1] + grades[0][2]
    124
```

List Operators[2]

Accessing list elements using using the **slicing operator**

```
>>> class_grades = [72,100,52,89,77,81,44,82,90]
>>> grades[0:2]
    [72, 100]
>>> grades[::-2]
    [90, 44, 77, 52, 72]
```

List Operators[3]

Modifying list elements using subscription

```
>>> my_list = ["apple", "banana", "pear"]
>>> my_list[1] = "plantain"
>>> my_list ['apple', 'plantain', 'pear']
```

Modifying list elements using the slicing operator

```
>>> class_grades[0:2] = [40, 45]
>>> class_grades
    [40, 45, 52, 89, 77, 81, 44, 82, 90]
>>> list1 = [[1, 2], [3, 4], [5, 6, 7, 8]]
>>> list1[2][1:3] = [11, 22, 33, 44]
>>> list1
    [[1, 2], [3, 4], [5, 11, 22, 33, 44, 8]]
```

List operators[4]

- Testing if a value is in a list using the **membership operator**

```
>>> 'asparagus' in ['apple', 'asparagus', 'broccoli']
    True
```

- Creating new lists using the **concatenation operator**

```
>>> list1 = ['apple', 'banana', 'pear']
>>> list2 = list1 + ['peach', 'orange']
>>> list2
    ['apple', 'banana', 'pear', 'peach', 'orange']
```

- Creating new lists using the **repetition operator**

```
>>> list1 = ['apple', 'banana', 'pear']
>>> list1 * 2
    ['apple', 'banana', 'pear', 'apple', 'banana', 'pear']
```

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Aliasing in Python

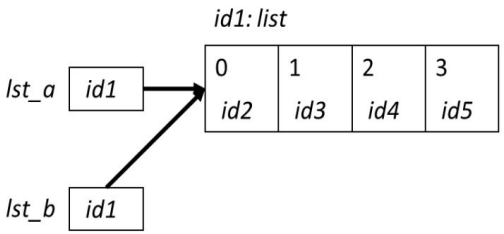
if nothing else, write #cleancode

Aliasing and Lists – 1

In Python, variables are just names that store references to values (aka objects).

⇒ When more than one variable refers to the same object, they function as **aliases** for that object.

```
>>> list = [0, 1 ,2 ,3]
>>> lst_a = list
>>> lst_b = list
```



- Sometimes we want to take advantage of ali
- Other times we need to avoid it.

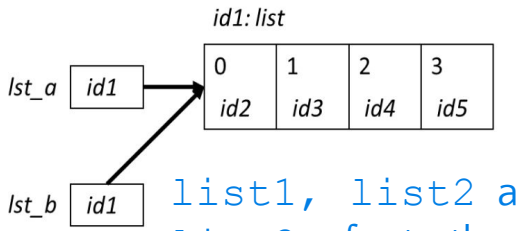
=> How do we avoid aliasing?

> By using a copy of the list of interest !

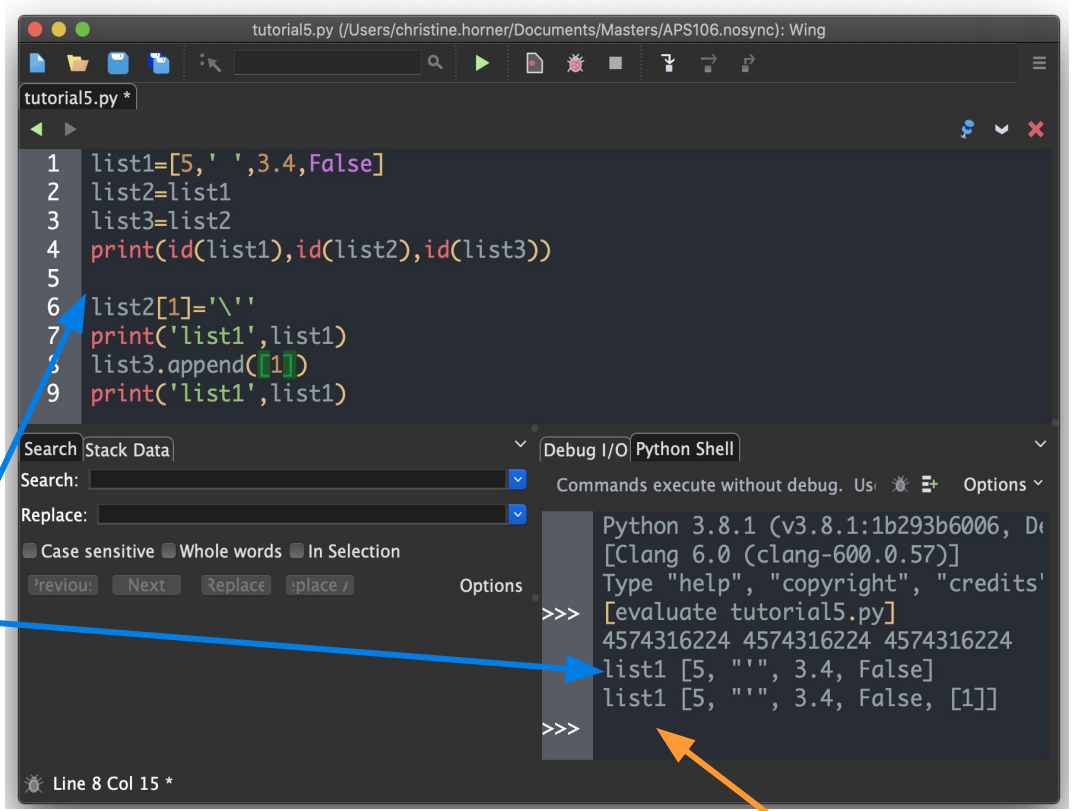
```
>>> list = [0, 1 ,2 ,3]
>>> lst_a = list[:]
>>> lst_b = list[:]
```

Aliasing and Lists -2

Modifying the object referred to by one of these variable will affect the other variables !!



`list1, list2` and `list3` refer to the same object



Changes made to `list2, and list3` are "visible" to `list1`

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Iterating over lists

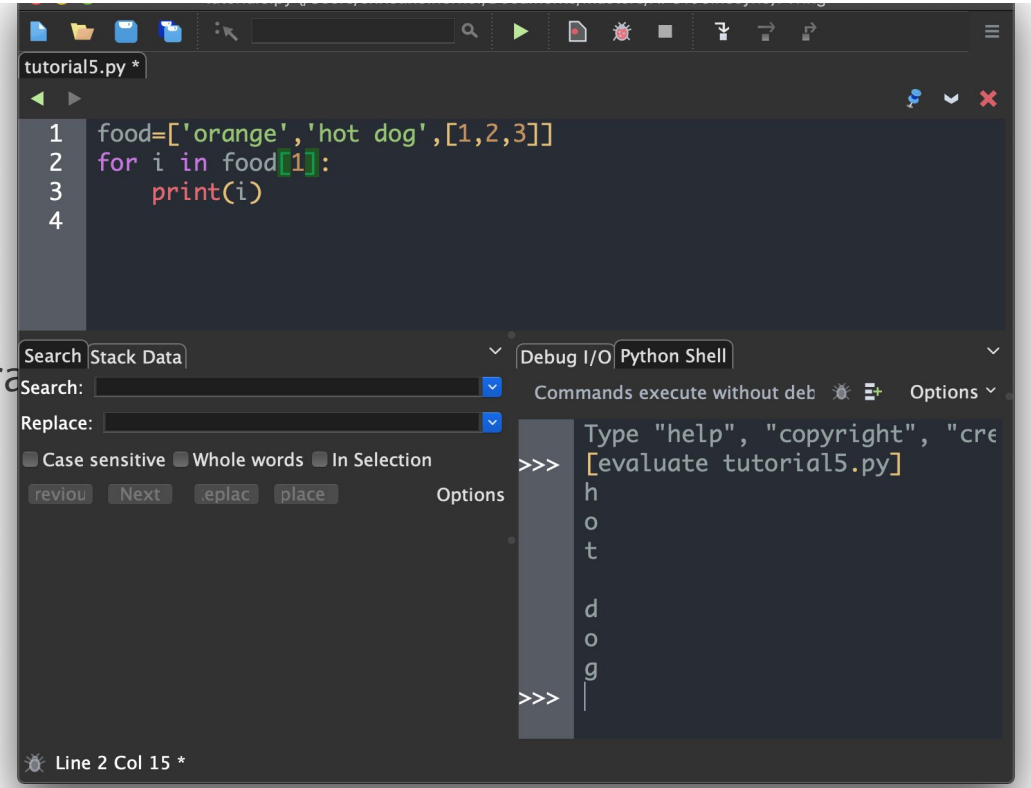
if nothing else, write #cleancode

Looping through a list

```
for list_item in list_object:
    loop_body
```

This syntactic pattern can be used to iterate over all Python sequences, e.g., lists, strings, etc.

⇒ each element in the sequence is "visited" only once



Practice Problems

slido

What is answer to the question below?

ⓘ Start presenting to display the poll results on this slide.

Review Practice Problem 1

Q1. What is answer to the question below?

The following list is created in Python:

```
my_list = ['hello', 'hey’]
```

Which of the options do NOT change the value of `my_list` to :

```
['hello', 'hey', 'hi']
```

<u>Note that multiple answers may be selected</u>

A. `my_list.append("hi")`

B. `my_list.append(["hi"])`

C. `my_list.extend("hi")`

D. `my_list.extend(["hi"])`

E. `my_list = my_list + "hi"`

slido

What does this code output?

ⓘ Start presenting to display the poll results on this slide.

Review Practice Problem 2

Q1. What does this code output?

```
x = [1, 2, 3]
y = [2, 4, 6]

print(x + 2 * y)
```

A. `[5, 10, 15]`

B. `[1, 2, 3, 4, 8, 12]`

C. `[1, 2, 3, 2, 4, 6, 2, 4, 6]`

D. `Error is thrown`

Coding Question 1

Write a function `sort_list()` that takes in a list of integers, sorts it in an ascending order, and returns the sorted list.
The function should not modify the original input list: that list must remain unchanged

Usage example:

```
>>> sort_list([48, 39, 1, 444, 39,2398])
    [1,  39, 39, 48, 444, 2398]
```

Coding Question 2

Write a program that repeatedly asks the user for grocery items (strings) until the user inputs ‘exit’, then display all their grocery items in alphabetical order.

Hint:

You can use question 1 to alphabetize the list

slido

Any Questions?

ⓘ Start presenting to display the poll results on this slide.