APS106

UNIVERSITY OF TORONTO

Tutorial 11 – Week 12

We'll be starting at the 10 minute mark

if nothing else, write #cleancode

APS106 Teaching Assistant Evaluations

- Please complete an evaluation for your teaching assistant(s)!

- Please use QR code
- Or the following link:

https://forms.office.com/r/xjf0aFbJ4k

Agenda

- Lab 7 review
  - `mol_form()` function
- Lecture review
  - Object-oriented programming review
  - Linked Data structures: Linked Lists
- Practice questions

Learning Objectives

After this tutorial, learners should be able to:

- recognize / describe / create Python classes
  - recognize / describe / create data attributes
    - recognize / describe / create class data attributes
    - recognize / describe / create instance data attributes
  - recognize / describe / create methods
    - recognize / describe / create class initializers (__init__)
    - recognize / describe / create non-initializer methods
- recognize / describe / create Python objects
- call methods on class / class instance objects
- recognize / describe / create linked data structures

Review of Lab

`mol_form()` function

Lab 7 review

```
def mol_form(compound_formula):
    """(str) -> dictionary
    When passed a string of the compound formula, returns a dictionary
    with the elements as keys and the number of atoms of that element as
values.

    >>> mol_form("C2H6O")
    {'C': 2, 'H': 6, 'O': 1}
    >>> mol_form("CH4")
    {'C': 1, 'H': 4}
    """
```

Hint: Use iteration to isolate the name of the element and its number. The following rules can be used:

- If a capital character is encountered, OR the end of the string is encountered, then it marks the end of the atom-number pair so you can start parsing the atom-number string and store it into the dictionary.
- If a digit character is encountered, mark it as the beginning of the atom number. Save this index since it will help you to split the string into the atom name and its number
- If a new atom starts but the digit character was not encountered for the previous atom, the number for the previous atom would be 1.

As always, the solution is not unique, and this is one of the many ways you can solve this question!

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Object-oriented programming review

Object Oriented Programming (OOP) Review

An object is a collection of related data and related functions

self refers to the instance of the class being manipulated

```
class Point:
    """ A 2d point, at coordinates x, y"""

    def __init__(self, x_arg = 0 y_arg = 0):
        """create a new point at x, y"""

        self.x = x_arg
        self.y = y_arg

    def distance_from_origin(self):
        """Compute my distance from the origin"""

        return (() + ()) ** 0.5
```

Data attributes

Method

APS106

UNIVERSITY OF TORONTO

Review of Lecture

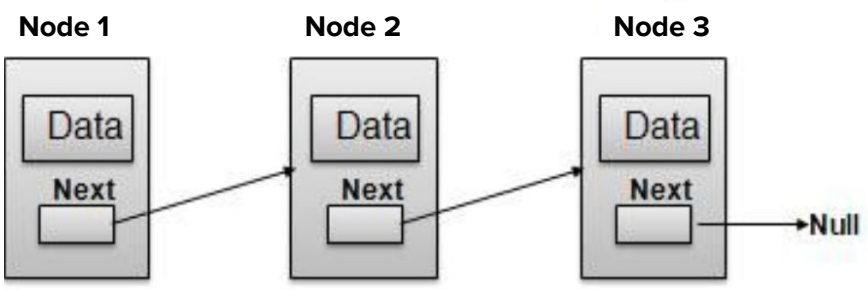
Linked Data structures: *Linked Lists*

if nothing else, write #cleancode

Linked Data Structures: Linked Lists

Collection of items (called **nodes)** each containing:

- data (called **cargo** in lecture)
- a **link** to the next item in the list



What's the Benefit of Linked Lists?

- The elements of a list can be stored in non-contiguous areas of memory (unlike arrays)
- **Inserting** and **removing** nodes does not require copying a large amount of memory
- Items remain organized (e.g., sorted) when nodes are inserted or removed

Operations on Linked Lists

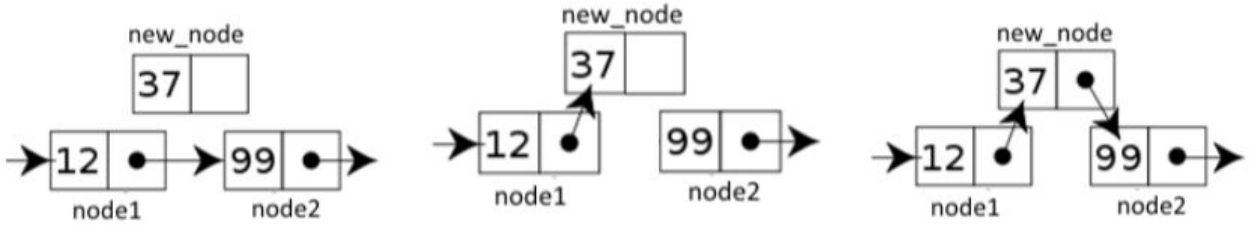
- **Insert** a new node (at the head/tail, at a specific position, etc.)
- **Delete/remove** a node (from the head/tail, from a specific position, etc.)
- **Update** a node (modify a node's value and/or link to the next node)
- **Search for/retrieve** a node with a given value
- **Search for/retrieve** a node with a given position in the list
- **Display/print** the list

Operations on Linked Lists – Insert

Sample insert operation: insert a new node at a particular position in the list

To insert a new node at a specific position:

- Go to/find the position in the list where the new node is to be inserted
- Update the node before the position (**node 1**) to point to the node you are inserting (**new_node**)
- Update the node you are inserting (**new_node**) to point to the node (**node 2**) previously linked to **node 1**

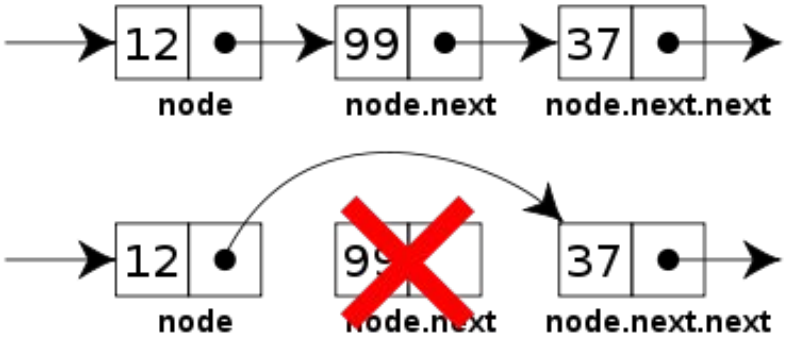


Operations on Linked Lists – Remove/Delete

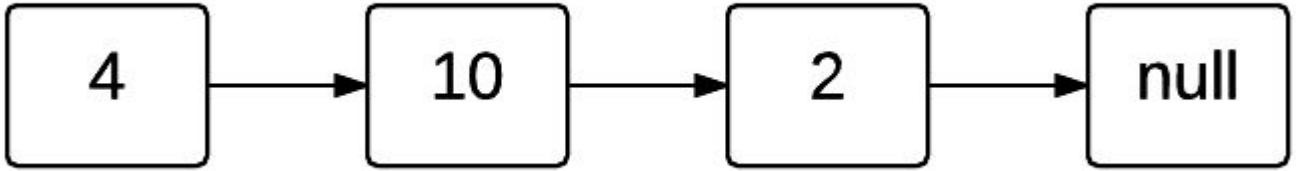
Sample delete operation: delete a node with a particular value

To delete a node with a given value

- Find the position in the list of the node to be deleted
- Update the previous node (**node 1**) to point to the node that the node you need to delete is pointing to.



How do you set-up a linked list?



Option 1: use a class Node (for list elements)

- We start by creating a **Node** class
- A node has two data attributes: **cargo** (i.e., the data/value of the node) and **next** (i.e., the link to the next node in the list)

```
class Node:

    def __init__(self, cargo_arg=None, next_arg=None):

        self.cargo = cargo_arg
        self.next = next_arg
```

Operations on Linked Lists

- **Insert** a new node (at the head/tail, at a specific position, etc.)
- **Delete/Remove** a node (from the head/tail, from a specific position, etc.)
- **Update** a node (modify a node's value and/or link to the next node)
- **Search for/Retrieve** a node with a given value
- **Search for/Retrieve** a node with a given position in the list
- **Display/Print** the list

Option 1: Inserting Elements

We can create a Linked List using Node Objects

```
# create three nodes and assign them to variables
node1 = Node('first')
node2 = Node('second')
node3 = Node('third')

# link the nodes together
node1.next = node2   #add node2 at the end of the list
node2.next = node3   #add node3 at the end of the list
```

Operations on Linked Lists

- **Insert** a new node (at the head/tail, at a specific position, etc.)
- **Delete/Remove** a node (from the head/tail, from a specific position, etc.)
- **Update** a node (modify a node's value and/or link to the next node)
- **Search for/Retrieve** a node with a given value
- **Search for/Retrieve** a node with a given position in the list
- **Display/Print** the list

Option 1: Printing all elements in a Linked List – 1

How do we produce a string representation of a list element (i.e., a Node object)?

We define the **__str__** method of class Node !

⇒ When we pass a **Node** object to **print(), print** will call method **__str__** of the object

to get its string representation. It will then print that string representation.

```
class Node:

    def __init__(self, cargo_arg=None, next_arg=None):
        self.cargo = cargo_arg
        self.next = next_arg

    def __str__(self):
        return str(self.cargo)
```

__str__ returns the string representation of the cargo of tha node

Option 1: Printing all elements in a Linked List – 2

To print all elements in a linked list, we will iterate through the list:

```
# create three nodes and assign them to variables
node1 = Node('first')
node2 = Node('second’)
node3 = Node ('third')

# link the nodes together
node1.next = node2
node2.next = node3

node = node1 # position the loop variable at the head of the list

while (node != None):  #move along the list until there are no more nodes
     print (node)
     node = node.next  #follow the link to the next node
```

Option 2: use a LinkedList Class

- While a linked list can be implemented using just the Node class, you can also create a class to represent linked lists.
- We start by creating a `LinkedList` class and defining the __init__ method

```
class LinkedList:

    def __init__ (self):

        self.length = 0  #attribute to keep track of the length of the list

        self.head = None #attribute to keep track of head node of the list
```

Operations on Linked Lists

- **Insert** a new node (at the head/tail, at a specific position, etc.)
- **Delete/Remove** a node (from the head/tail, from a specific position, etc.)
- **Update** a node (modify a node's value and/or link to the next node)
- **Search for/Retrieve** a node with a given value
- **Search for/Retrieve** a node with a given position in the list
- **Display/Print** the list

Option 2: Insert elements – 1

Sample insert operation: add an element at the head of the list

```
class LinkedList:

    def __init__ (self):
        self.length = 0  #attribute to keep track of the length of the list
        self.head = None #attribute to keep track of head node of the list


    def add_first(self, cargo_arg):
        ''' Add an element with cargo cargo_arg as the first item of the list '''
        node = self.head
        new_node = Node(cargo_arg, node)
        self.head = new_node
        self.length += 1
```

Copy in a helper variable the head of the list

Create a new node with the value to be inserted.

Make the new node the head of the list

Increment the length of the list by 1

Option 2: Option 2: Insert elements – 1

Sample insert operation: add an element at the tail/end of the list

```
class LinkedList:

    …
      def add_last(self, cargo_arg):
          ''' Add an element with cargo cargo_arg as the last item of the list '''
          node = self.head
          new_node = Node(cargo_arg)

          while node.next != None:
               node = node.next

          node.next = new_node
          self.length += 1
```

First item of linked list

Create new Node for the item to add to the list

Find the last element of the list

Link the last element to the new node

Increment the length of the list by 1

APS106

UNIVERSITY OF TORONTO

Practice Problems

Review Practice Problem 1

Q1. In Python, a function within a class definition is called:

A. a program
B. an operation
C. a method
D. an attribute
E. an object

slido

In Python, a function within a class definition is called:

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

Review Practice Problem 1

Q1. In Python, a function within a class definition is called:

A. a program
B. an operation
C. a method
D. an attribute
E. an object

Review Practice Problem 2

Q2. What does the following code output?

```
1   class Node:
2
3       def __init__ (self, cargo = None, next = None):
4           self.cargo = cargo
5           self.next = next
6
7       def __str__(self):
8           return str(self.cargo)
9
10  node1 = Node("one")
11  node2 = Node("two")
12  node3 = Node("three")
13
14  n = node1
15  while n!= None:
16      print(n, end = ', ')
17      n = n.next
```

A. `one, two, three`
B. `one, two, three,`
C. `one,`
D. An error is thrown
E. None of the above

slido

What does the following code output?

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

Review Practice Problem 2

Q2. What does the following code output?

```
1   class Node:
2
3       def __init__ (self, cargo = None, next = None):
4           self.cargo = cargo
5           self.next = next
6
7       def __str__(self):
8           return str(self.cargo)
9
10  node1 = Node("one")
11  node2 = Node("two")
12  node3 = Node("three")
13
14  n = node1
15  while n!= None:
16      print(n, end = ', ')
17      n = n.next
```

A. one, two, three
B. one, two, three,
C. one,
D. An error is thrown
E. None of the above

Review Practice Problem 3

Q3. If a node n is the last item in a linked list, what should the next_node attribute in n be assigned to?

A. null
B. none
C. None
D. 0

slido

If a node n is the last item in a linked list, what should the next_node attribute in n be assigned to?

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

Review Practice Problem 3

Q3. If a node n is the last item in a linked list, what should be the value of its next_node attribute?

A. null
B. none
C. None
D. 0

Review Practice Problem 4

Q4. If a linked list is empty, which of the following statements is true?

A. head is None
B. tail is 0
C. head != tail
D. head < tail

slido

If a linked list is empty, which of the following statements is true?

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

Review Practice Problem 4

Q4. If a linked list is empty, which of the following statements is true?

A. head is None
B. tail is 0
C. head != tail
D. head < tail

Review Practice Problem 5

Q5. What does the following code output?

```
1   class Node:
2
3       def __init__ (self, cargo = None, next = None):
4           self.cargo = cargo
5           self.next = next
6
7   node1 = Node("one")
8   node2 = Node("two")
9   node3 = Node("three")
10  node1.next = node2
11  node2.next = node3
12
13  n = node1
14  while n!= None:
15      print(n, end = ', ')
16      n = n.next
```

A. `one, two, three`
B. `one, two, three,`
C. `one,`
D. An error is thrown
E. None of the above

slido

What does the following code output?

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

Review Practice Problem 5

Q5. What does the following code output?

```
1   class Node:
2
3       def __init__ (self, cargo = None, next = None):
4           self.cargo = cargo
5           self.next = next
6
7   node1 = Node("one")
8   node2 = Node("two")
9   node3 = Node("three")
10  node1.next = node2
11  node2.next = node3
12
13  n = node1
14  while n!= None:
15      print(n, end = ', ')
16      n = n.next
```

A. `one, two, three`
B. `one, two, three,`
C. `one,`
D. An error is thrown
E. None of the above

Coding Question 1

1) Let's work together to create a linked list of the items you ate as meals yesterday (eg, your breakfast, lunch and dinner).

2) Once we've made the list, add a new node representing an afternoon snack and insert it into the linked list.

3) Then print out all your meals for the day.

Coding Question 2

Use the starter code given for the LinkedList class to implement the following methods:

- `__str__` : should return a string representation of all the nodes in the list (with the same format Python uses to convert regular lists to strings).
- `__len__` : should return, as an integer, how many nodes are in the list
- `insert` : should work the same way as the insert method for the Python built-in type list. *Hint: Use help (list.insert) in Wing101 for more info.*
- `concatenate` : takes in a LinkedList object and adds its elements to the end of the self object. Should work the same way as the concatenation operation for the Python built-in type list.

slido

Any questions?

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