**Work done**

15th – 17th April

1. map() – used for mapping a function to an iter object
2. .split() method for indexing elements separated by white spaces.

The results are put into a list. An object can be specified within the split(“s”), hence it will split spaces and the ‘s’ within the object being splitted

The opposite of split() is join. To join a splitted variable a with “-“, we write (“-”).join(a)

Things to note about the split() method:

* If the split parameter is not specified, it will use a white space. For a string with multiple consecutive white spaces, this will remove all white spaces.
* If the split parameter is specified to a white space, such as split(“ ”), it will remove only one white space for multiple consecutive white spaces

1. Sum(a) for items in a if a is an iter object like list, dictionary or tuple. The items in the object should be integers
2. Precise handling- specifying number of decimal places of float. There are different ways of doing this:

* round(float, no.of.decimal.places) eg round(i, 2)
* print(“%.2f” % i)

18th April

1. the os module provides functions for directly interacting with the operating system

for example, os.environ outputs a dictionary with keys representing an attribute of the operating system, and the values representing the values of those attributes

to print the OS of the computer, we can write print(os.environ[“OS”]), and for my laptop, it’ll output … Windows\_NT

1. The following are useful string methods to differentiate:
2. .upper() or .lower() that changes all items of the string to uppercase or lower case respectively
3. .capitalize() that changes only the first item of the string to uppercase
4. The open( ) is a file handling function that puts a file into a variable as a read, write, append or create method

For a text file awesome.txt, we can open it for reading as

>>> f = open(“awesome.txt”, “r”)

“r” is for “read”, “w” is for “write”, “a” is for “append” and “x” is for “create”

1. Variables inside a print() function can be aligned:

* Print(‘Kevin’.ljust(20, ‘-’)
* Print(‘Kevin’.rjust(20, ‘-’)
* Print(‘Kevin’.center(20, ‘-’)

To print variables in line, use a comma to separate them e.g

Print(‘Kevin’, ‘ ‘, ‘Obuya’)

This will output

… Kevin Obuya

*BONUS*

The os module also has a system function for manipulating the system of your machine/device/laptop. For example, to shutdown your laptop, we can write:

os.system(“shutdown /s /t 0”)

The /s can be replaced with /r for restarting instead of shutting down. The /t describes the amount of time in seconds after which the function will be run. This code can be used to schedule your system for certain functions such as shutting down your computer after a given amount of time, let’s say if you feel you might fall asleep while watching a movie, hence the laptop switches itself off.

19th April

1. The textwrap module.

This is used for formatting or wrapping text with a given width.

If I was to wrap the text “hflspvgwjute” which has 12 character, with a width of 4, the output would return “hfls”, “pvgw”, “jute”.

>>> N = “Kevin”

>>> print(textwrap.TextWrapper(width=1).wrap(text=N))

. . . [“K”, “e”, “v”, “i”, “n”]

>>> print(textwrap.TextWrapper(width=1).fill(text=N)))

. . . K

. . . e

. . . v

. . . i

. . . n

1. Assigning multiple variables at the same time is done in different ways:
2. >>> x, y = 4, 5
3. >>> x, y = input().split() # this converts the input into a list then assigns the values accordingly
4. Strings can be aligned according to a certain with to form shapes.

We can use the .ljust(), .rjust() and .center() as earlier learnt in 18th April above

1. Strings can be concatenated using the + operator. How ever, if we want to remove some items in a string, we can use .replace() function and replace them with an empty string “”. This method takes two parameters: first the character to be replaced, followed by the item being used to replace it. A third parameter can be added to specify how many times the replacement will happen.

Addidtionally, we can use the .lstrip() or .rstrip(), then specifiy the characters being removed from the left or right

1. We can change integers into binary, octal or hexagonal values using the bin(), oct() and hex() methods respectively.

They all return the values preceded by the definition of the state of the integer.

For example:

>>> print(bin(2)) # the binary form of 2 is 10

. . . 0b10

>>> print(hex(2)) # the hexadecimal form of 2 is 2

. . . 0x2

>>> print(oct(2)) # the octal form of 2 is 2

. . . 0o2

20th April

1. The rangoli (Indian folk art)



21st April

1. 

25th April

1. The itertools module.

Python’s Itertool is a module that provides various functions that work on iterators to produce complex iterators. This module works as a fast, memory-efficient tool that is used either by themselves or in combination to form iterator algebra.

Functions found in the itertools module include:

1. count

>>> from itertools import count

>>> for i in count(start=1, step=2):

>>> if i > 10:

>>> break

>>> print(i)

… 1

… 3

… 5

… 7

… 9

1. product
2. permutations

This function returns all available permutations of an iterable object. It uses the iterable object as the first argument, and the length of each permutation as the second argument. If the length of each permutation is not specified, it uses the length of the list in the first argument.

1. A counter is a container that stores elements as dictionary keys, and their counts are stored as dictionary values. It’s found within the collections module

26th April

1. The Counter() class in the collections module returns a dictionary from a list. The dictionary has a value as the keys and the number of occurrences of that value as the dictionary values.

>>> my\_list = [1,2,1,3,4,5,4,3,1,4]

>>> print(Counter(my\_list))

… {1:3, 2:1, 3:2, 4:3, 5:1}

We can hence use dictionary methods on the resulting dictionary created by the Counter() class

We can get the keys or values of the keys using the keys() and values() methods respectively.

This method does not return a type list. It returns a type dict\_keys or dict\_values respectively

<class 'dict\_keys'>

<class 'dict\_values'>

1. Important list methods:
2. The pop() method removes a particular index. If the index is not specified, it removes the last one
3. The remove() method removes a specified item in the list, and it requires the item being removed as an argument
4. The clear() method clears the list, making it empty
5. The insert(index, value), which requires two arguments, the index where the value is being inserted and the value itself. This then pushes the rest of the items in the list further up the index.
6. The extend() method that adds a list into another list

Thislist.extend(another\_list)

1. Etc
2. The cmath module

A polar coordinate is a complex version of a cartesian coordinate.

A complex number is written in the format x + yj where x is a real number and y is an imaginary number(a real number that exists as a multiple of j)

A polar coordinate is represented like (r, ø), where r is the magnitude of the line from the coordinate point to origin, and ø is the angle between the line and the x-axis calculated anti-clockwise.

r is computed using abs(polar coordinate)

ø is computed using phase(polar coordinate)

27th April

1. Defaultdict is a container like dictionaries present in the module collections. Defaultdict is a sub-class of the dictionary class that returns a dictionary-like object. The functionality of both dictionaries and defaultdict are almost same except for the fact that defaultdict never raises a KeyError. It provides a default value for the key that does not exists.
2. To print an iter object horizontally, we can use print(\*list) for list as the iter object
3. The calendar module provides classes and functions that deal with operations involving the calendar. We will deal with it more later

29th April

1. The weekday() function under the calendar module takes the year, month and day as integer arguments and returns the index of the day of the week of that given date.

The indexes of the days of the week start with Monday as index 0

1. Error exceptions are used to give an alternative feedback if an error is raised.

For example, dividing an int by 0 gives a ZeroDivisionError.

Instead, we can tell the system what to do when such an error is raised



1. Ø

30th April

1. Sets are an unordered collection of unique values. A single set contains values of any immutable data type.

Sets are very weird. They change their arrangement each time the code initializing them is run.

Consider two sets, A and B.

Union of these sets is a set containing values that exist in A or B without repeatables

Intersection of sets is a set containing values that exist in both A and B

Difference of sets is a set containing values that exist in A and not in B, or B that are not in A



1. The .add() method adds its parameter/argument into a random position of the set
2. OR operations are represented by the .union() method, and AND operations are represented by the .intersection() method
3. The symmetric\_difference() method returns a set containing elements in A that are not in B and those in B that are not in A
4. Itertools combinations() and combinations\_with\_replacement() methods
5. The “”.join() method

1st May

1. From collections import deque(). This method allows for list methods to be applied from a certain side, like .appendleft(), popright()

2nd May

1. Regex or regular expression is a sequence of characters used for defining a pattern. This pattern could be used for searching, replacing and other operations. Regex is extensively utilized in applications that require input validation, Password validation, Pattern Recognition, search and replace utilities (found in word processors)

import re

1. In the re module, we have the compile method that is used to check if a given string contains valid re expressions
2. Palindrome.

3rd May

1. The product function from the itertools module

4th May

1. all() and any()
2. made a tic tac toe game all on my own, and posted it on github.

5th May

1. the zip() function. This returns an object of tuples of elements of the same index of iterables.

>>> zip([1,2,3],[4,5,6],[7,8,9])

…. (1,4,7), (2,5,8), (3,6,9)

This would have been an amazing tool in making the tic tac toe game yesterday

1. The lambda function is an anonymous function. This means it doesn’t have a name.

The normal way of defining a function is by using the “def” keyword.

Lambda function s can be embedded into a variable.



1. The map() function is used to apply a function through items of an iter.
2. Different ways of printing list items:



7th May

1. Let’s learn a bit about classes:

The \_\_init\_\_() function is used to assign values to object properties, or other operations that are necessary to do when the object is being created.

The \_\_str\_\_() function defines the string representation of objects formed from the class.

1. The split() function is used to split() a string using a certain parameter such as white-spaces. The function only takes one argument. For several arguments, we can use the split() function under the re module, with a predefined variable containing the multiple split factors.
2. A filter takes a function returning True or False and applies it to a sequence, returning a list of only those members of the sequence where the function returned True. A Lambda function can be used with filters.

The map() and filter() functions work the same way, except the fact that the filter() uses functions that return Booleans.

1. The reduce() functions from the functools module also acts like the map() or filter() functions, except it takes two args and iterates through the list in doubles from left to right.

8th May

1. The Fraction class within the fractions module takes in two integers as args and returns a fraction representation with the first arg being the numerator and the second being the denominator
2. Let’s look at the re module

https://www.guru99.com/python-regular-expressions-complete-tutorial.html

A regular expression specifies a set of strings that matches it. The functions in this module let you check if a string matches a particular string pattern.

In a re pattern, there are specific characters within the r””(raw input) normally used:

* \w to such for any alphanumerical characters, the opposite being \W
* \s to match any whitespaces, the opposite being \S
* \d to match any digits, the opposite being \D

The re module has several functions such as:

1. The split() function.

In this module, the split() function acts in the same way as we’ve done before, except for the fact that it can take more than one args

1. The findall() function

This function returns a list of pattern matches upon a string.

1. The search() function

This function searches a string for a pattern match

1. The match() function

This functions only searches for a match in the first character of a string

9th May

1. We have learnt a little about regex search patterns

[] is used to enclose search patterns that are grouped together

^$ denote start with and end with of string respectively

11th May

1. Regular Expressions

Matching metacharacters of regular expressions include:

* The []

These square brackets are used to enclose multiple characters

[A-Z] can be used to represent all upper-case alphabet letters.

[rgd] can be used to represent either ‘r’, ‘g’ or ‘d’

* The \*, + and ?

The \* suggests that a character should appear zero or more times

The + suggests that a character should appear one or more times

The ? suggests that a character should appear 1 or 0 times

If we say r”ca\*t”, the pattern can be used to search “ct”, “cat”, or “caaat”, for an finite number of “a”.

Instead of these, we can specify the number of times to search a character using curly braces {}. ca{1,2}t will match “cat” and “caat” only

12th May

1. p.sub(“str”, h, count=1). This expression substitutes any pattern p matches from h with “str”, once as defined by the count

13th May

1. when we use (string literals) as the [1:] part of a regex, why doesn’t it print the first part?????

Encountered problem at ‘Hex Code’ in ‘preparation’ github repo

1. Introduction to html.parser module. To use the HTMLparser class that’s within the html.parser module, we create a sub-class of it.

After this, we can use its functions against html code to retrieve start tags, end tags, data within tags, empty tags, comments and more information

I feel like this is a good way of scrapping data from web pages.

14th May

1. commonly used HTMLParser methods include:
2. handle\_starttag(self, tag, attr)
3. handle\_endtag(self, tag)
4. handle\_startendtag(self, tag, attr)
5. handle\_comment(self, data)
6. handle\_data(self, data)

15th May

Monthly Review

* learned a lot by doing rather than reading or going through w3schools content
* problem solving still taking too long
* algorithmic approach proves to speed up things
* productivity decreases a lot during debugging
* should include all possible test occurrences in initial algorithm design
* current weaknesses are regular expressions and class methods

16th May

What next?

* Continue with problem solving
* Work on speed, efficiency and clean code design using algorithms and comments
* Continuously review previous works
* Start some Django
* Continue to contests or advanced challenges if completed problem solving
* Work on weaknesses

17th May

1. Did and miraculously passed the first two parts of the turing.com vetting process
2. The first test was about general stuff in python was challenging. It consisted of topics such as threading and multi-threading, socket programming, Unicode, best practices and code optimization, classes and inbuilt class functions
3. The second test a rather easy coding challenge that was similar to the ones I have encountered in Hackerrank.com
4. Now I remain with the practical challenge and possibly the face-to-face interview

18th May

1. Today we have to finalize on regular expressions especially in the use of ‘?’

*Backreferencing*

In regex, we use () to specify a certain group of literals to be matched. These groups can be accessed using the group() method.

group(0) always returns the whole match

group(1) returns the first group of matches in the regex

if we want to use, let’s say the first group of matches, at a current position, we use \1

*Named Groups*

Instead of using backreferencing to refer to groups, we can also name the groups using

(?P<name>). Afterwards, these named groups can be used inside of a regex pattern using (?P=name), or in a sub() method as the substitution string using (?g<name>)

*Look around*

This is used to check for certain expressions or patterns before or after a pattern

For look behind we use (?<=B)A or (?<!B)A

For look ahead we use A(?=B) or A(?!B)

In the above statements, B is the pattern to check if it comes before or after the matching pattern A.

One thing to not is, whenever we use ( or ) for grouping in a regex, any other parts of the expression will be used to match a string but will not be included in the find all. This is because it considers group() as all the items in the literals between the ()

We hence need to use (?:B) to counter the grouping

I think we’ve now finalized everything on regular expressions

19th May

1. Let’s look at decorators in python

Functions in python can be used dynamically by:

1. Assigning them to variables.

In this case, the () is not used during assignment since that will call the function. Hence, we only assign the function name



1. Assigning them as a parameter of another function



1. Assigning them as the return value of another function



In decorators, functions are passed into other functions as arguments then called as wrapper functions

We’ll do more about the advanced topic of decorators

1. numpy methods include:
2. numpy.array()
3. numpy.shape(array)
4. numpy.reshape(array, (new shape))
5. numpy.transpose(array)
6. array.flatten()
7. numpy.concatenate((array1, array2), axis=0)
8. numpy.zeros((rows, collumns), dtype=datatype)
9. numpy.ones((rows, collumns), dtype=datatype)
10. numpy.eye(rows, collumns, k=0) k is 0 by default
11. numpy.identity(num)
12. numpy.sum() and numpy.prod()
13. numpy.min() and numpy.max()
14. numpy.mean/var/std ()
15. numpy.dot/cross (A, B)
16. numpy.inner/outer(A, B)

20th May

1. what is the difference between a python module, package, and library?

A module is a python file (.py) that has functions and global variables that can be executed individually.

A package is a directory that contains a collection of modules and contains a \_\_init\_\_.py file by which the interpreter reads it as a package.

A library is a collection of related functionalities of code that can be reused without actually writing code. It is often considered as a collection of packages

A constructor is a function of a class that is called whenever an object is created. It’s purpose is to assign values into the object of that class. For example, \_\_init\_\_.py

1. A method is a function that is associated with an object

Three types of methods are:

1. Instance methods
2. Class methods
3. Static methods

22nd May

1. Let’s go through the xml.etree.ElementTree module

XML stands for extensible markup language and it’s a mark up language just like html.

Extensible Markup Language is a markup language and file format for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

We can use the xml parse from a string as follows

>>> root = xml.etree.ElementTree.fromstring(xml string)

root.tag returns the parent tag of the xml

to get child tags, we can iterate through root.

1. Now I have finally gained some understanding on decorators.

In my own words, a decorator is a function that takes a function as args. Within it, we create another function that takes care of calling the function used as args within the decorator. Let’s call the inner function “fun”. So, fun takes care of calling the function being decorated. The function being decorated is created outside the decorator.

When we use the @wrapper function above the outside function, this means that when the outside function is called, it will first go through some edits within the decorator function before being implemented.

25th May

A regular expression is a sequence of characters that describes a search pattern.

It is mainly used for string pattern matching



The dot(.) matches anything but a new line

\d matches anything that is a digit( [0-9] ). \D matches anything that is not a digit

\s matches any whitespace characters [ \r\n\t\f ]:

1. \r is carriage return
2. \n is new line
3. \t is tab
4. \f form feed

Other characters are \b for backspace

https://stackoverflow.com/questions/35928774/what-do-b-f-n-r-do-what-are-the-differences

\S matches any non-whitespace characters

^ and $ match beginning and ending of a string respectively

{x, y} repetitions match x to y times, like a range

[] encloses a match pattern which can be a range such as [A-Z] for uppercase letters only

[^B] matches anything but B such as [^A-Z] for anything but an uppercase letter

26th May

() is used for grouping patterns

? is used to match zero or one times

29th May

\b is word boundary. If we say \bh\b, a match will only be found if h is between a word boundary or at the beginning or end of a sentence. A word boundary is anything but a word(\W)

30th May

If we need to read an input as stdin, we use the sys module and use the sys.stdin.read() function

31st May

Socket programming is a way of connecting two sockets or nodes in a network in order to communicate with each other. One socket is created to listen for communication (the server) while another is created to reach out for connection (the client).

When we click on a link, our browser does something like the following:



When the browser does this, it creates a socket s that can be used to request a page. The same socket can now be used to receive the requested information before being destroyed. A socket is destroyed after completing its task.

We can say that the above is what happens in the client socket.

What happens in the server socket is different.



In the above picture, we first create the server socket. After that, we use the bind method which takes an argument as a tuple containing the host name and the port being used.

The hostname can be as above or 127.0.0.1(localhost) but in this case, it will only be visible to the same machine.

If we specify the hostname as an empty string, it means the server socket is reachable by any address. Lower ports are usually reserved for “well known” services such as HTTP or SNMP.

Finally, the argument to listen() tells that the maximum number of requests the server socket can have waiting in a queue. 5 is the normal max.

Now that we have a server socket listening on port 80, we can enter the main loop of the web server:



This is all a server socket does. It doesn’t send or receive any data. It just produces “client” sockets.

Let’s continue on how to use sockets tomorrow.

2nd June

The client-server architecture refers to a system that hosts, delivers and manages most of the resources and services that the client requests. The client is responsible for making requests, and receiving the response. The server is responsible for receiving requests from clients and sending responses back to the clients.

When creating a socket, whether a server or a client, we use the socket module.

The socket.socket() method takes two arguments, first the address family, then the protocol.

The address family, in most cases, is AF\_INET, which represents IPV4 addresses. It can be AF\_INET6 for IPV6, AF\_BLUETOOTH, and more.

The protocol is normally SOCK\_STREAM which represents TCP (transmission control protocol).

It can be SOCK\_DGRAM for UDP (user datagram protocol), or others

We’ll delve more into sockets later

5th June

Managed to finish all regex problems from hackerrank. I have now killed one weakness from Month 1

Another weakness from last month is classes, which I have also gotten the basics. Now to dive deeper into using decorators with class methods.

When creating a child class from an existing parent class, we can override the init() function from the parent class by creating a new init() function in the child class. In this case, other functions of the parent class are still inherited by the child class.

6th June

Abstraction is an important feature of OOP. It is the ability to hide implementation details from the user. The abc module in python helps in creating abstract classes (abstract base class)

From the abc module, we can use the ABC abstract class or ABCMeta for creating abstract classes and the abstractmethod for creating abstract methods within our created abstract classes.

8th June

Let’s talk about linked lists.

A linked list is a data structure that contains nodes that have data and pointers to the next node. The elements or nodes of a linked list are not stored contiguously (in an ordered fashion).

Each node has a pointer to the next node.

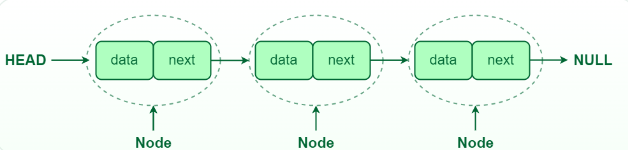
The first node of the linked list is called the head node, and has data and a pointer to the next node.

The last node of the linked list is called the tail node and has a pointer which is null since it doesn’t point to any other node.

There are three types of linked lists:

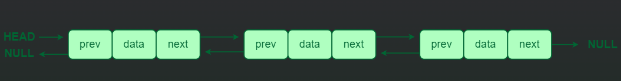
1. Single linked lists:

In this type, the nodes contain only one pointer which points to the next node



1. Double linked lists:

In this type, the nodes contain two nodes. One of the points to the next node while the other points to the previous node

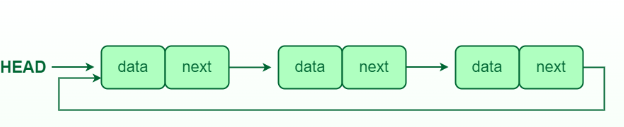


1. Circular linked lists:

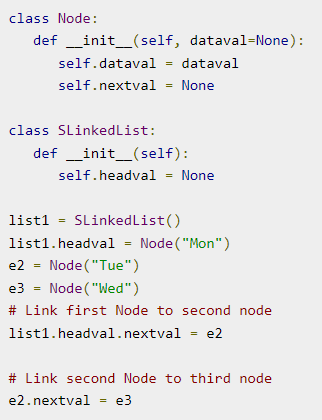
This type can either be single or double. There is no need for a head node in this type.

The last node points to the first node if it is a single circular linked list.

The first node points backwards to the last node if it is a double circular linked list.



Linked lists can be created in the following way:



First, we define the Node class which will take care of creating nodes. Afterwards, we create a class SLinkedList for the head after which we create an object for it (list1). At this point, list1 is the head node but it does not have a pointer to the next node, and has no data.

Next, we give the list1 head node a value (headval), and make it an object of the Node class with a datival variable. At this point, list1.headval is a node, with a value(“Mon”), but still has no pointer to the next node since we haven’t created any other nodes.

To create other nodes, we create objects of the Node class as seen with e2 and e3.

At this point, we have three nodes, list1.headval with a value “Mon”, e2 with a value “Tue” and e3 with a value “Wed”.

Now, to create the pointers for each node, we first start with the head node list1.headval.

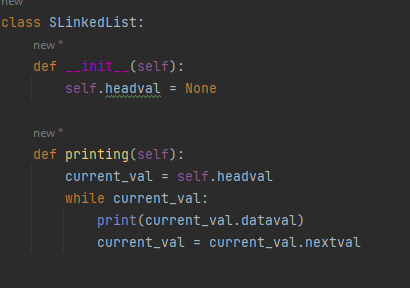
We simply do this by giving list1.headval.nextval as e2. At this point, list1.headval has a pointer to the next value, which is e2. We do the same for e2 as e2.nextval = e3.

In order to print the values of the linked list, we can create a new function under the SLinkedList class that prints the datival and iterates through the list by updating the print variable to nextval.

9th June

*Traversing through a Linked list*

In order to traverse through a linked list and print its items, we need to define a new function for printing the linked list items.

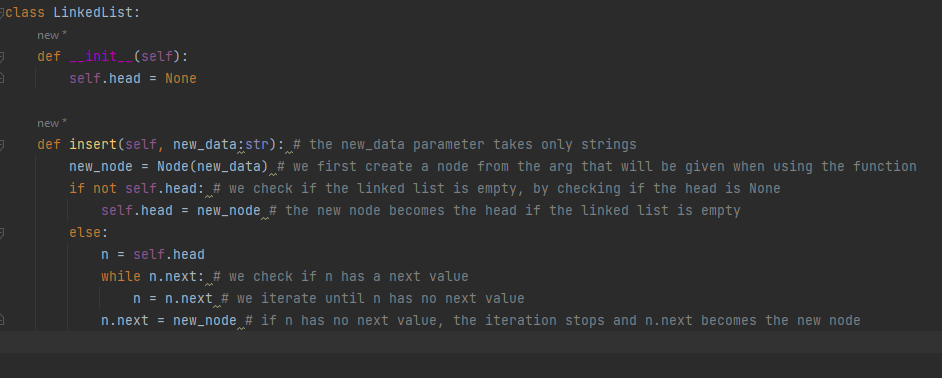


In the above code snippet, first we create a new variable current\_val that stores the head node (headval). We then start an iteration for printing the data in current\_val, as we continuously update the value of current\_val to its next node. When we reach the tail node, which has a None value as its nextval, the iteration stops.

From the two code snippets above, we have manually added nodes by defining a new node as the next value of the previous node. We need to define a function to add new nodes as the come in.

*Adding items at the end of a Linked list*

Let us first create a new code snippet to describe the addition of new nodes.

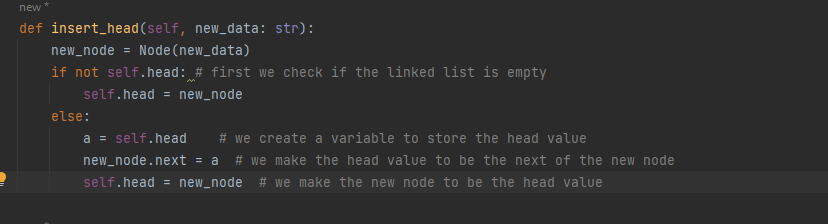


So now we have created an insert function that takes in a string which will be used to create a new node (new\_node) within it. Afterwards, we check if the linked list is empty by checking if there is a head node. If there is no head node, then the new node becomes the head node.

If there if a head node, then we first create a new variable (n) which holds the head. In order to know if we have reached the tail node, we check if the current node we are in has a next value of none. So, we start an iteration that continues so long as the next of n is not none.

The iteration stops when the next value of n is not none. So, at this point, we give the next value of n as the new node.

*Adding items at the beginning of a linked list*



I think it’s all self-explanatory as seen from the comments.

*Adding items at a point within the linked list*

First, we create a new function for adding a node at a certain point of the linked list.

This function needs to take two arguments. The first will be the node being added, and the second will be the node after which to insert the new node.

Within the function, we iterate through the linked list until we reach the node after which we want to add the new node. The rest is explained in the comments.

*Deleting items from a linked list*

Deleting items from a linked list involves changing the next values of items in the linked list.

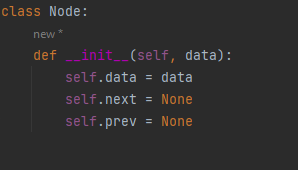
If we want to delete the head node, we can make the second node to be the head node.

If we want to delete the tail node, we can give a none value to the second last node

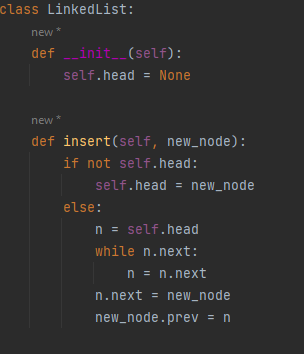
If we want to delete a certain node somewhere within or in the middle of the linked list, we change the next value of the node before the one we want to delete to point to the node after the one we want to delete.

So far, we have only dealt with singly linked lists.

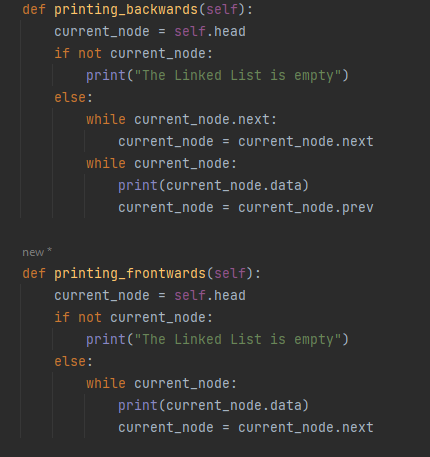
Since double linked lists have nodes that contain addresses to the next and previous nodes, we can add a previous (prev) into the node definition as shown below. In this case, the linked list can also be printed backwards.



As we add new nodes, we also define the previous of the added nodes.



So now we can traverse through the list backwards and frontwards.



We can’t traverse a single linked list backwards since it has no previous address.

As for circular linked lists:

* if it’s singular, then the tail node points to the head node.

This means, when inserting new items, we have to define where we are going to insert them. When inserting new items at the end, we can check whether the current node points to the head node.

* If it is double, the previous of the head node points to the tail node. When inserting new nodes into this type, we have to modify previous and next addresses of the prevailing and next nodes of the inserted node

Traversing through circular linked lists may require a different method. We can append a list with node items as we traverse the circular linked list then stop when we find that the current node during traversal is in the created list.