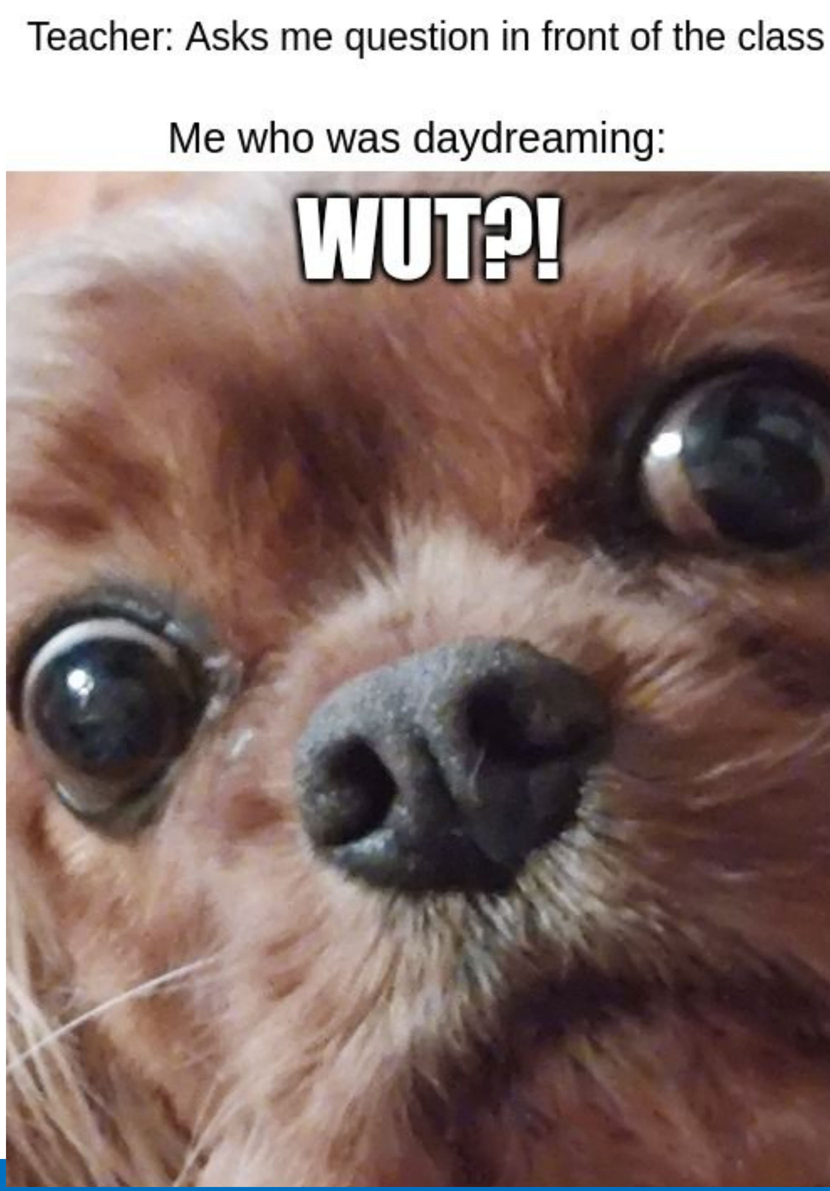
Unit II: Understanding Basic Data Types & Packages

Programming Methodology (CSF101)

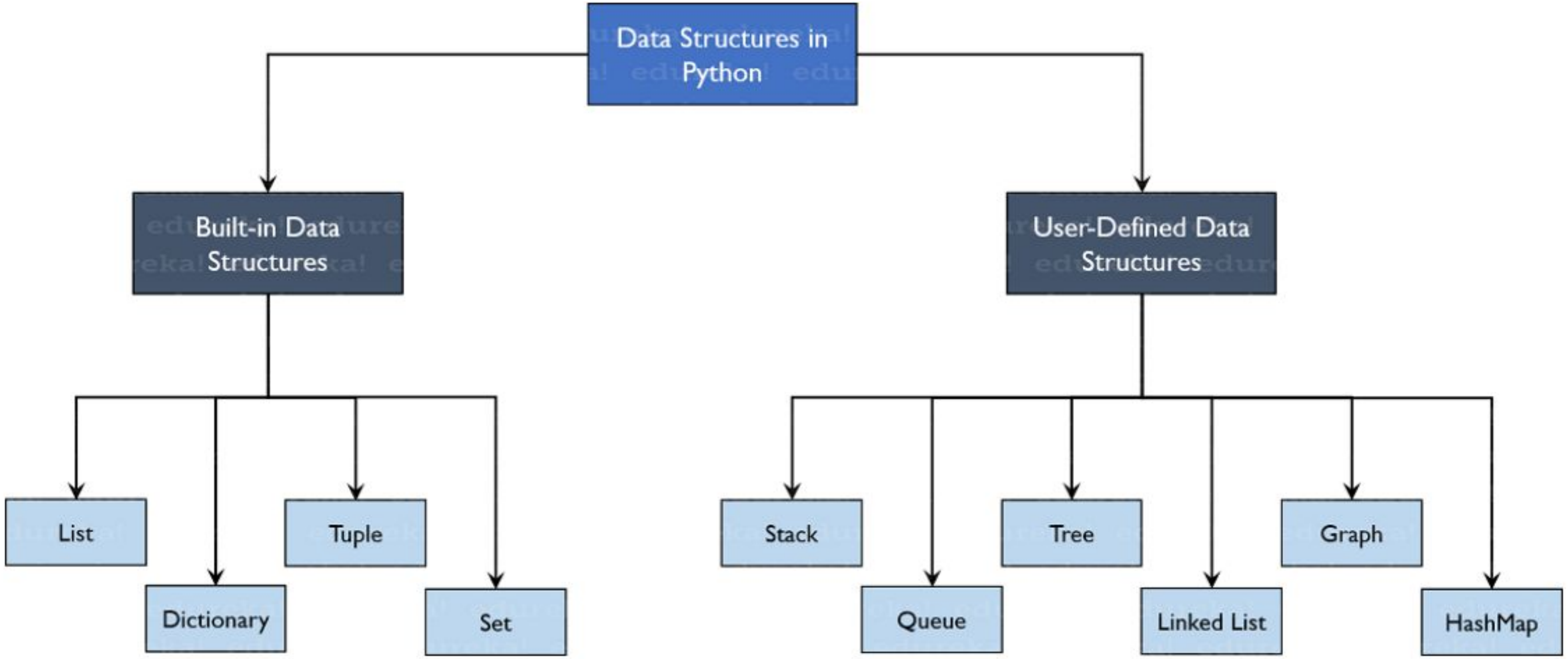
Royal University of Bhutan

Outline

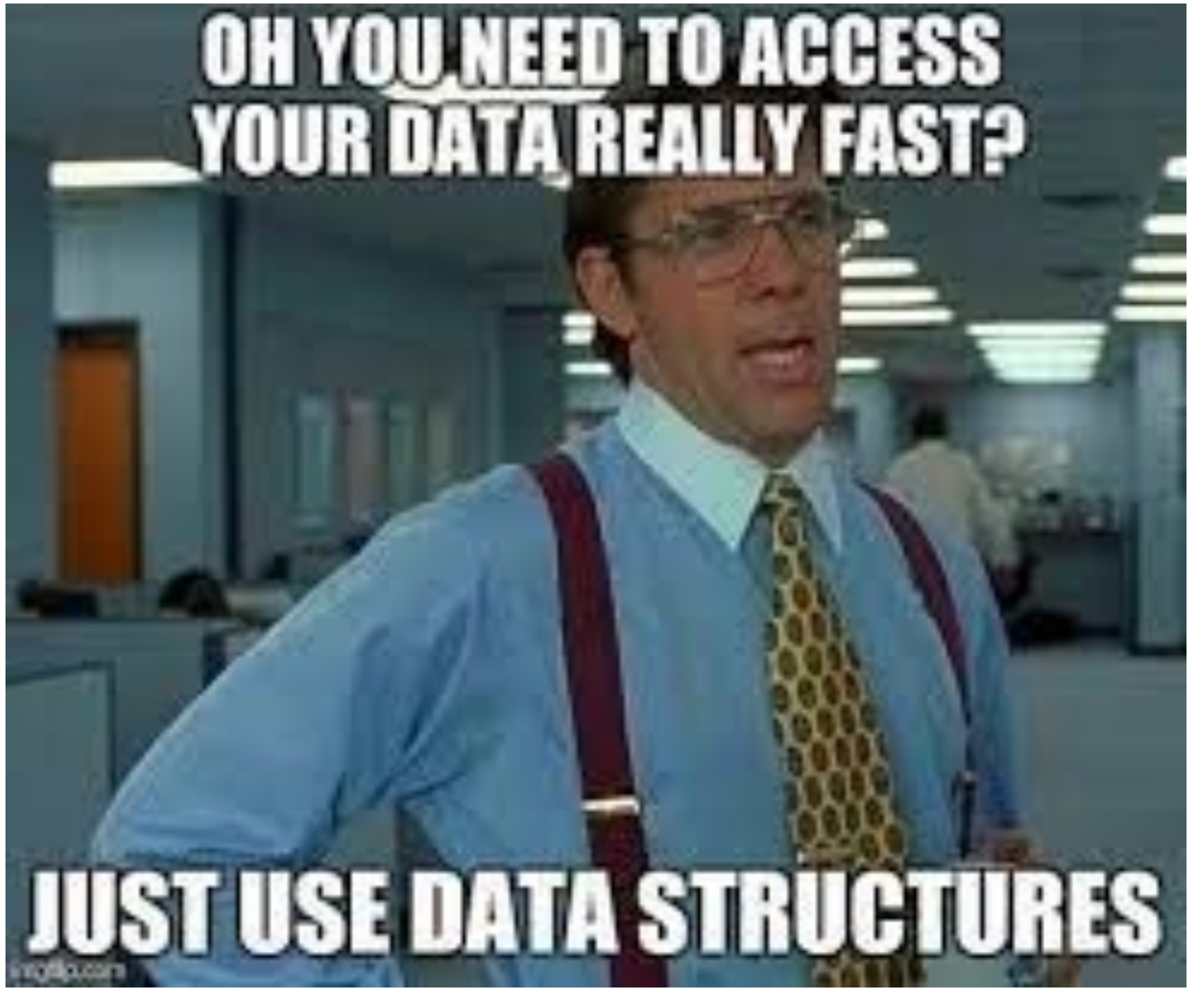
- Arrays and Multi-Dimensional Arrays
- Abstract Data Structures
- Standard & Third Party Language Packages



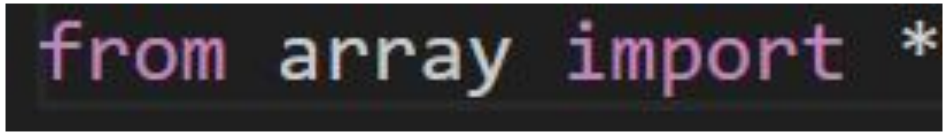
Data Structure



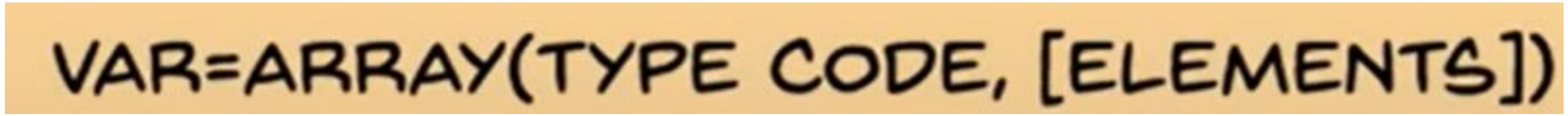
Cont…


OH YOU NEED TO ACCESS YOUR DATA REALLY FAST?
JUST USE DATA STRUCTURES
imgflip.com

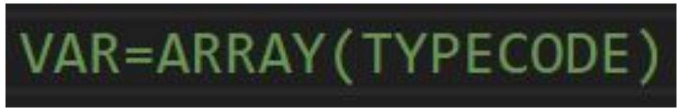
Array



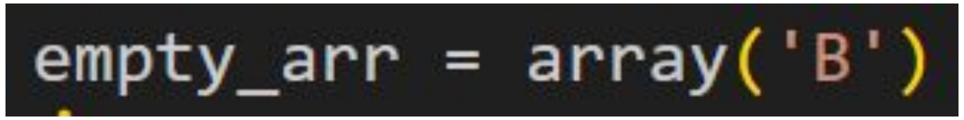
```
from array import *
```



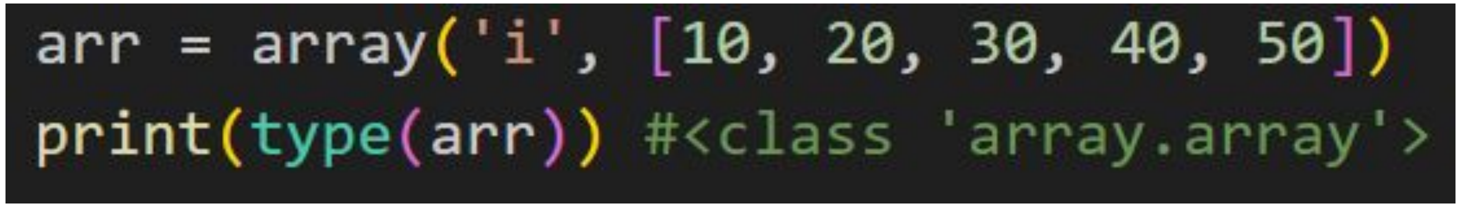
```
VAR=ARRAY(TYPE CODE, [ELEMENTS])
```



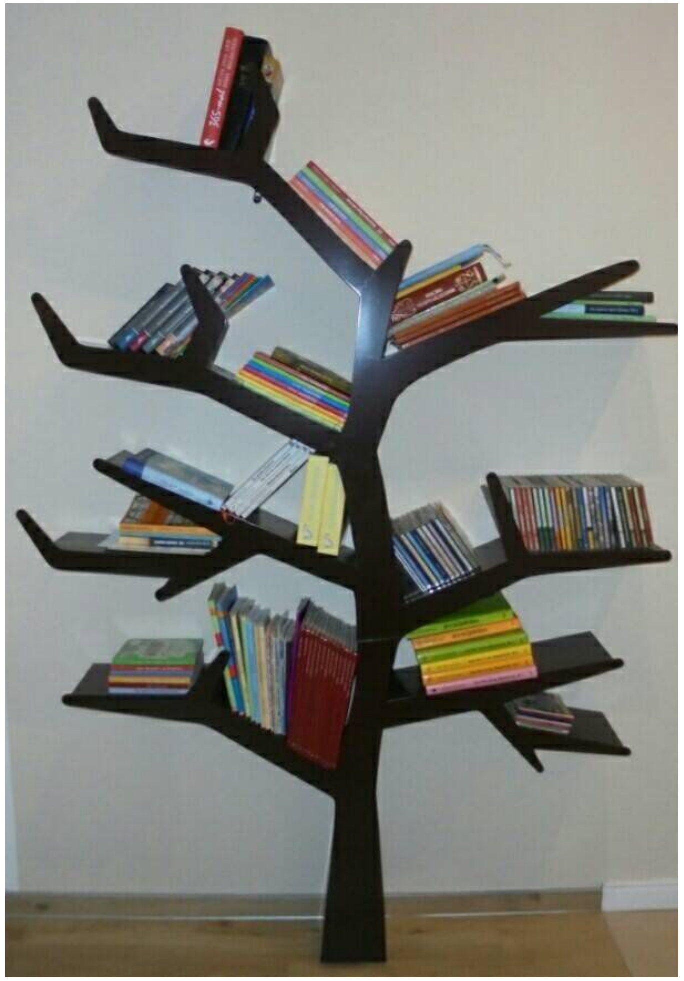
```
VAR=ARRAY(TYPECODE)
```



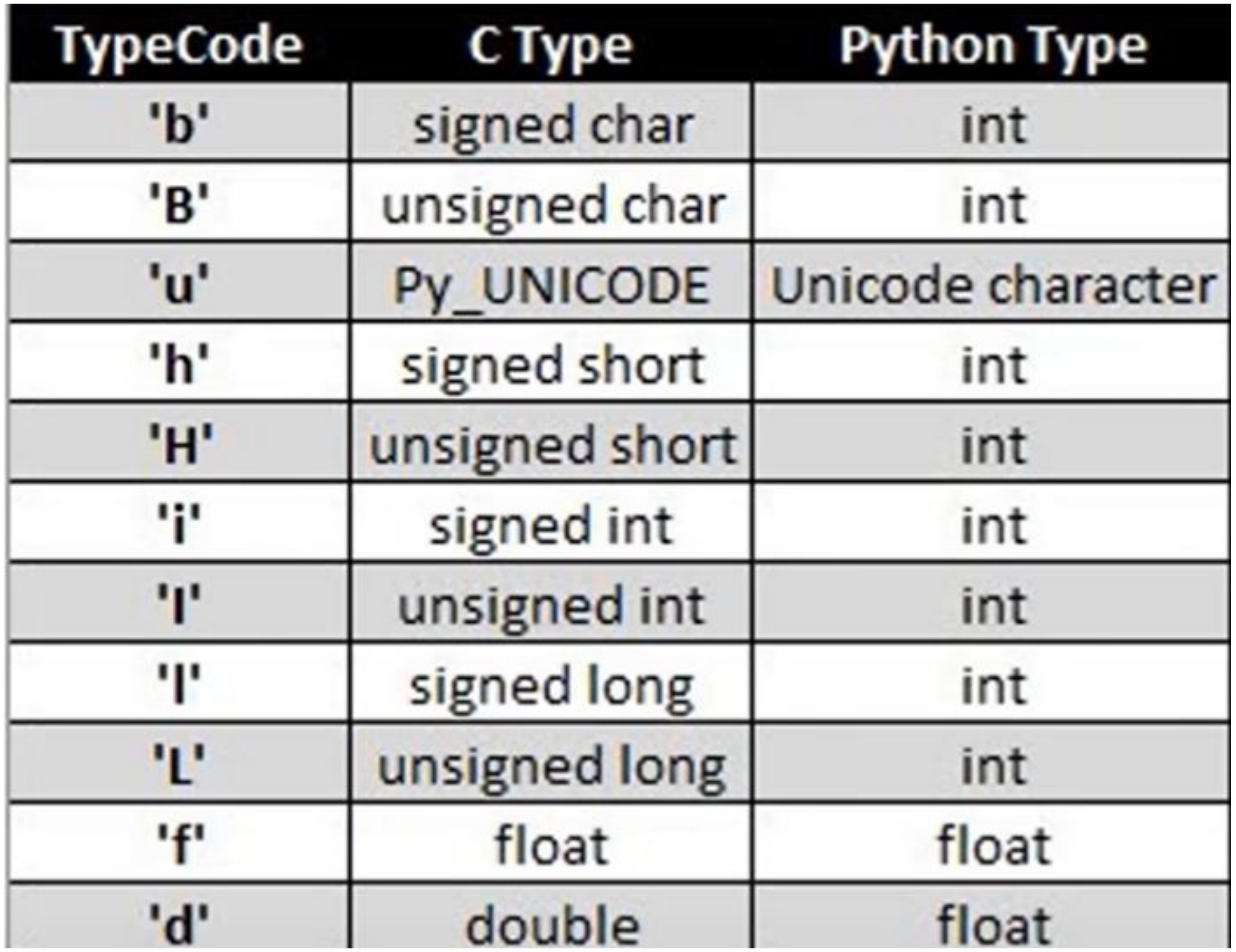
```
empty_arr = array('B')
```



```
arr = array('i', [10, 20, 30, 40, 50])
print(type(arr)) #<class 'array.array'>
```

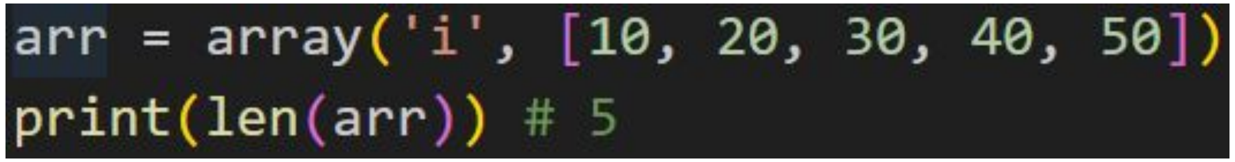


Typecode

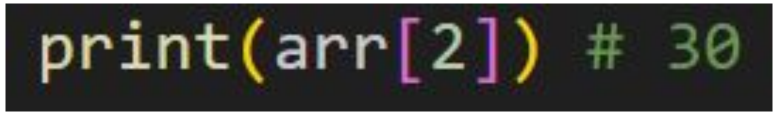


TypeCode	C Type	Python Type
'b'	signed char	int
'B'	unsigned char	int
'u'	Py_UNICODE	Unicode character
'h'	signed short	int
'H'	unsigned short	int
'i'	signed int	int
'I'	unsigned int	int
'l'	signed long	int
'L'	unsigned long	int
'f'	float	float
'd'	double	float

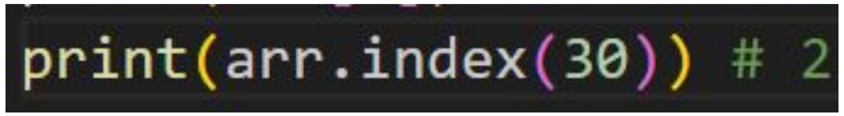
Methods in array



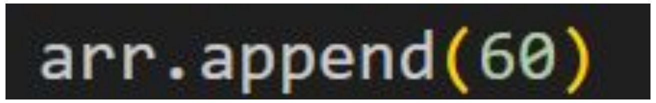
```
arr = array('i', [10, 20, 30, 40, 50])
print(len(arr)) # 5
```



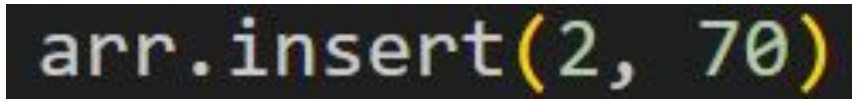
```
print(arr[2]) # 30
```



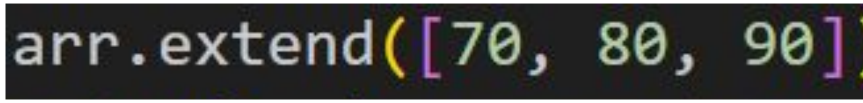
```
print(arr.index(30)) # 2
```



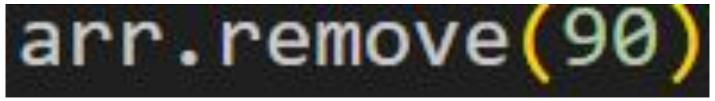
```
arr.append(60)
```



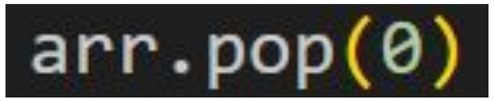
```
arr.insert(2, 70)
```



```
arr.extend([70, 80, 90]
```

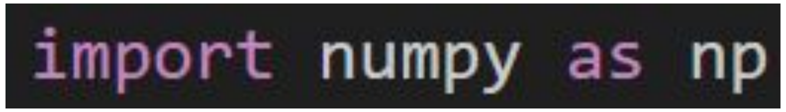


```
arr.remove(90)
```

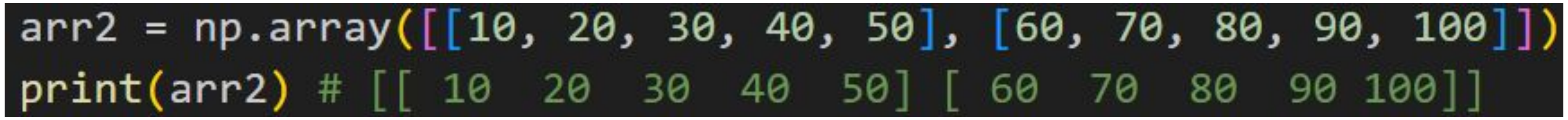


```
arr.pop(0)
```

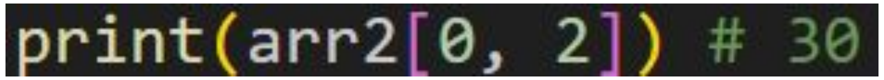
Multi-DImensional Arrays



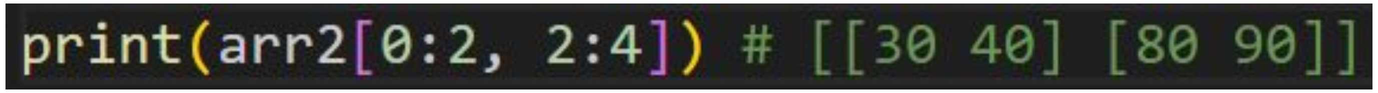
```python
import numpy as np
```



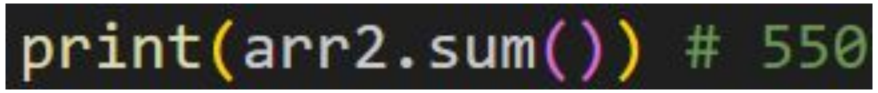
```python
arr2 = np.array([[10, 20, 30, 40, 50], [60, 70, 80, 90, 100]])
print(arr2) # [[ 10  20  30  40  50] [ 60  70  80  90 100]]
```



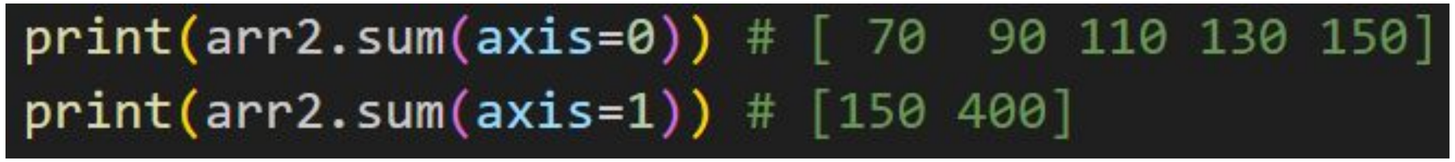
```python
print(arr2[0, 2]) # 30
```



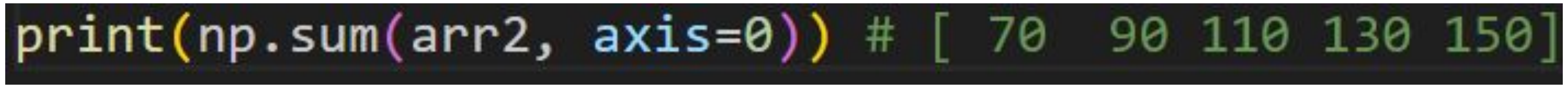
```python
print(arr2[0:2, 2:4]) # [[30 40] [80 90]]
```



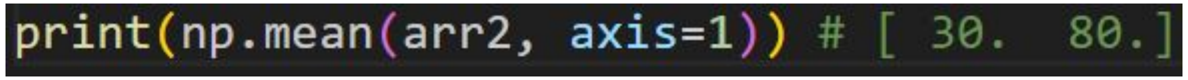
```python
print(arr2.sum()) # 550
```



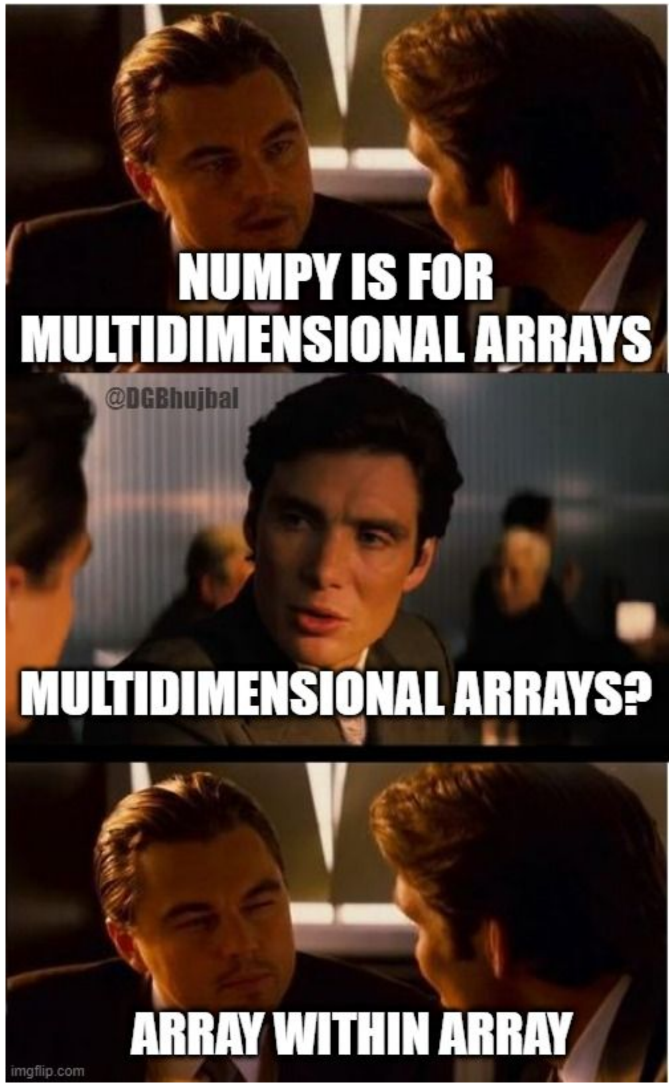
```python
print(arr2.sum(axis=0)) # [ 70  90 110 130 150]
print(arr2.sum(axis=1)) # [150 400]
```



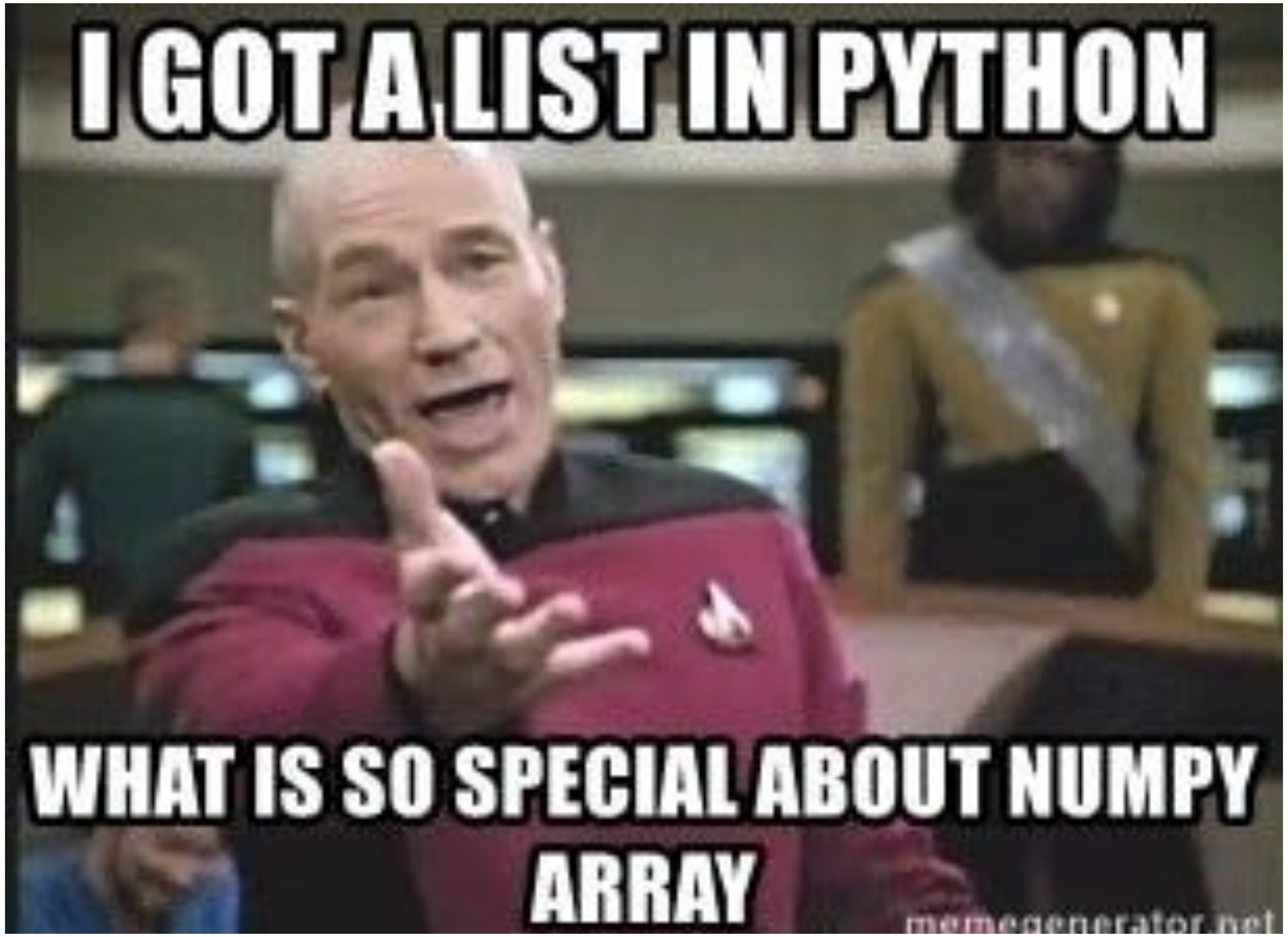
```python
print(np.sum(arr2, axis=0)) # [ 70  90 110 130 150]
```



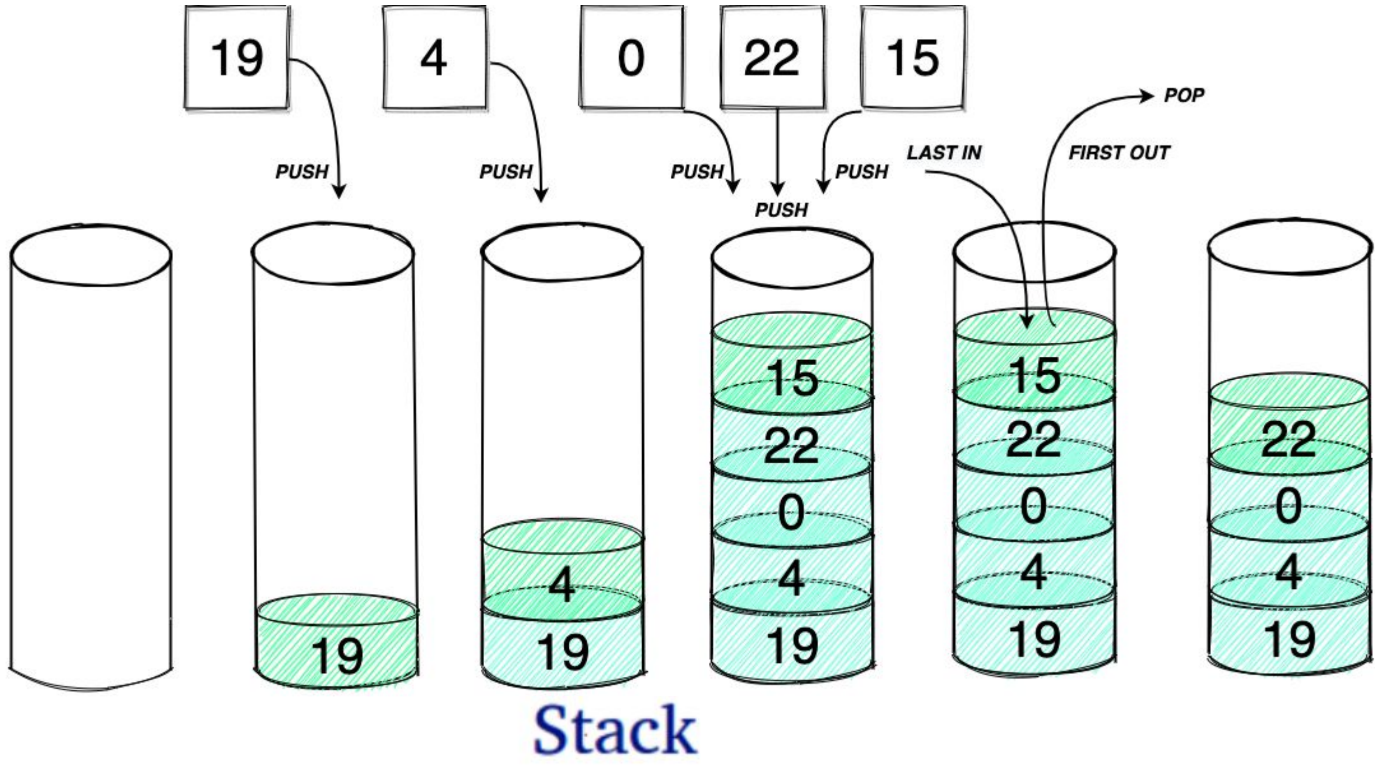
```python
print(np.mean(arr2, axis=1)) # [ 30.  80.]
```



Cont…


I GOT A LIST IN PYTHON
WHAT IS SO SPECIAL ABOUT NUMPY ARRAY
memegenerator.net

Stack

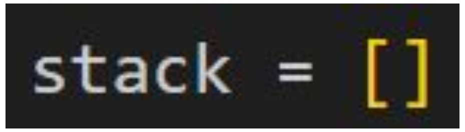


Insertion and Deletion happen on same end

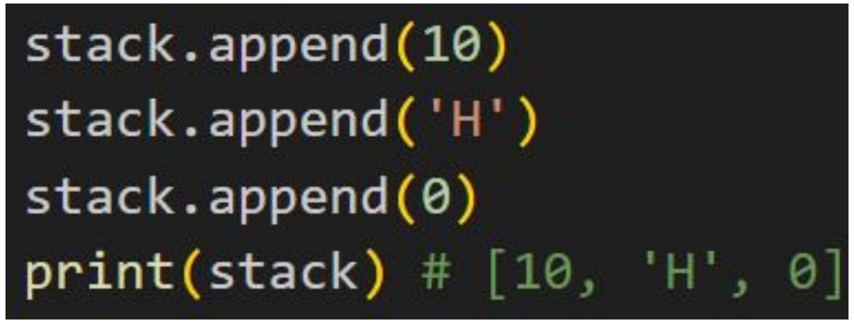
Functions associated with Stack

- empty() – Returns whether the stack is empty
- size() – Returns the size of the stack
- top() – Returns a reference to the topmost element of the stack
- push(a) – Inserts the element ‘a’ at the top of the stack
- pop() – Deletes the topmost element of the stack

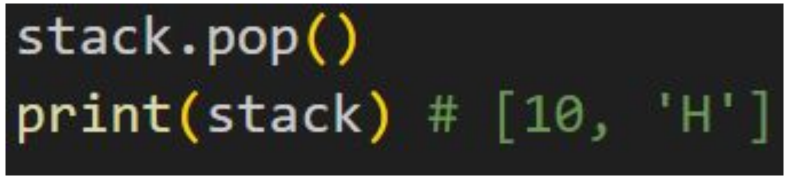
Implementing stack using lists



```
stack = []
```



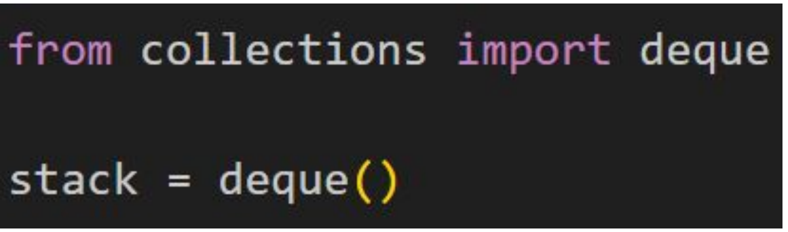
```
stack.append(10)
stack.append('H')
stack.append(0)
print(stack) # [10, 'H', 0]
```



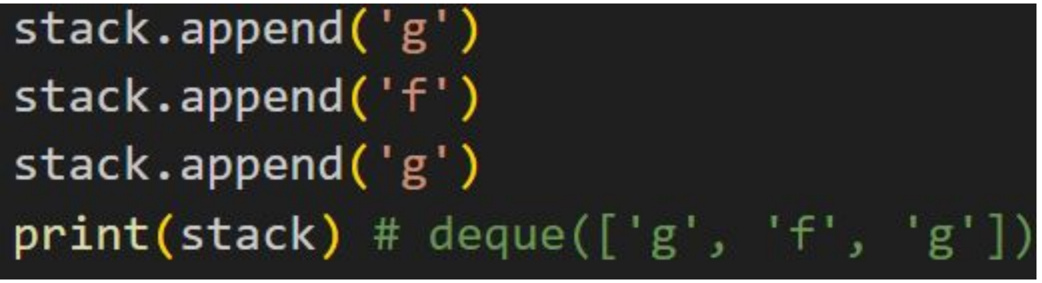
```
stack.pop()
print(stack) # [10, 'H']
```

STACK
IN PYTHON
USING LIST

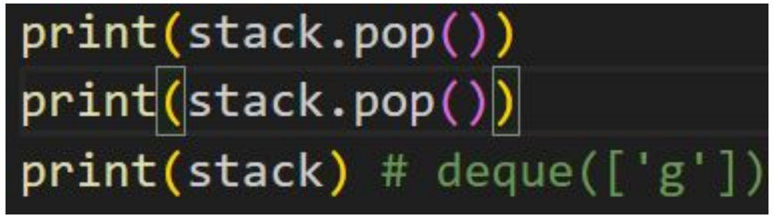
Implementing stack using dequeue & queue



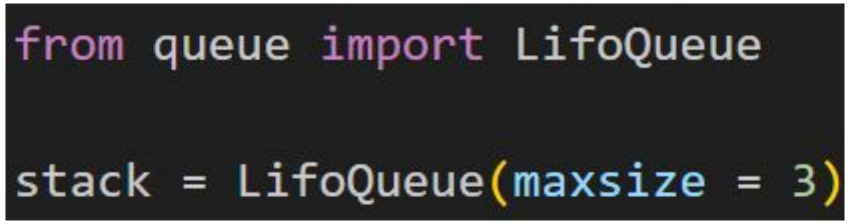
```python
from collections import deque

stack = deque()
```



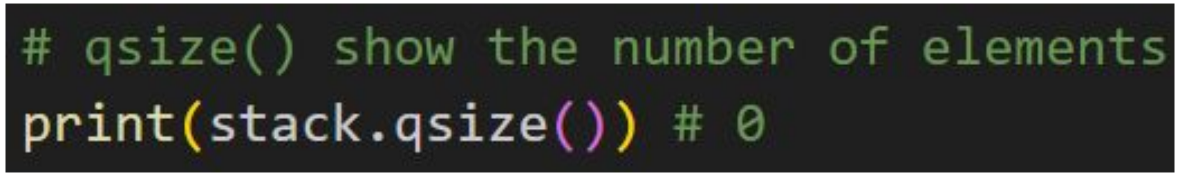
```python
stack.append('g')
stack.append('f')
stack.append('g')
print(stack) # deque(['g', 'f', 'g'])
```



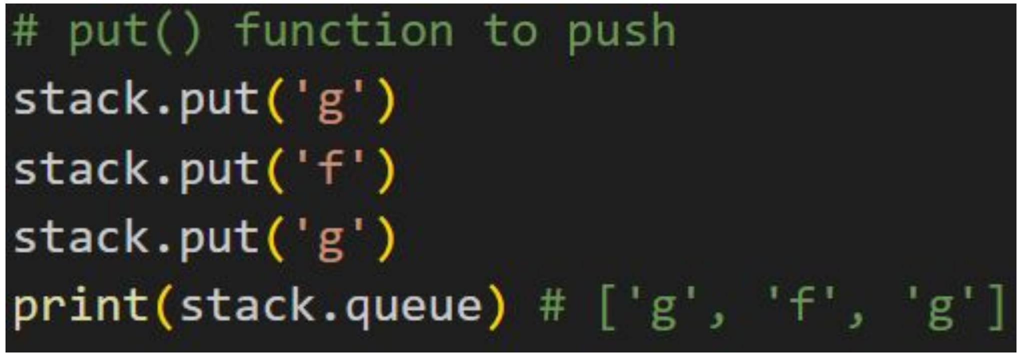
```python
print(stack.pop())
print(stack.pop())
print(stack) # deque(['g'])
```



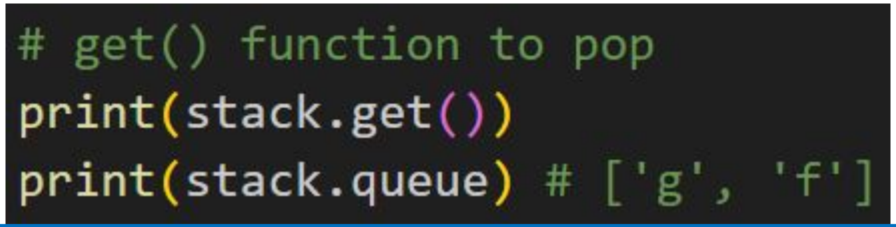
```python
from queue import LifoQueue

stack = LifoQueue(maxsize = 3)
```



```python
# qsize() show the number of elements
print(stack.qsize()) # 0
```

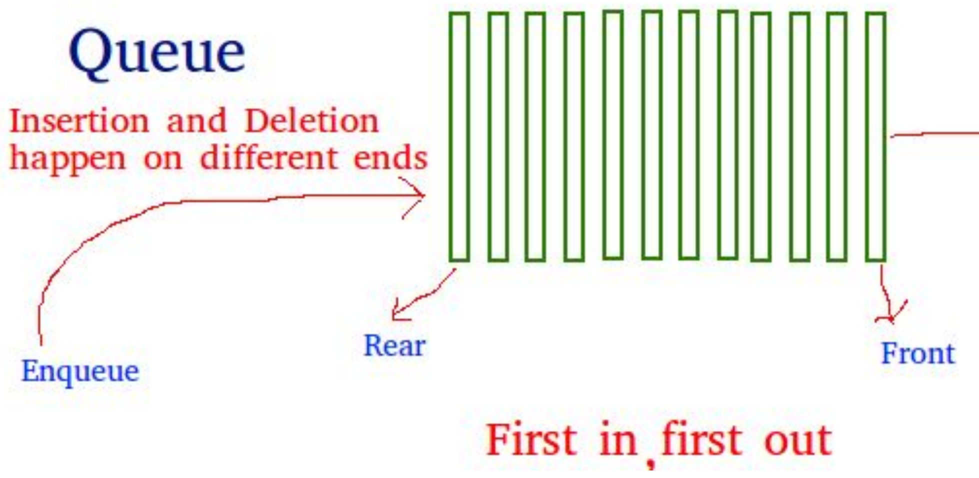


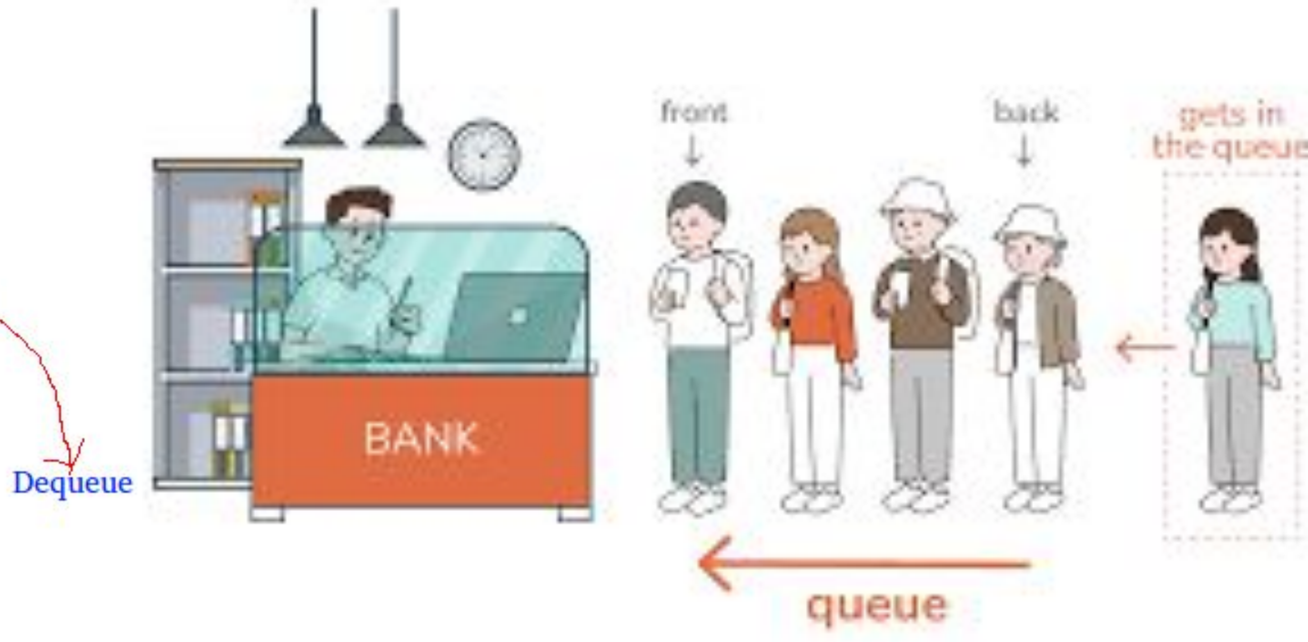
```python
# put() function to push
stack.put('g')
stack.put('f')
stack.put('g')
print(stack.queue) # ['g', 'f', 'g']
```



```python
# get() function to pop
print(stack.get())
print(stack.queue) # ['g', 'f']
```

Queue

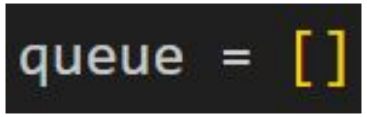

Queue
Insertion and Deletion happen on different ends
Enqueue
Rear
Front
Dequeue
First in first out


Dequeue
BANK
front
back
gets in the queue
queue

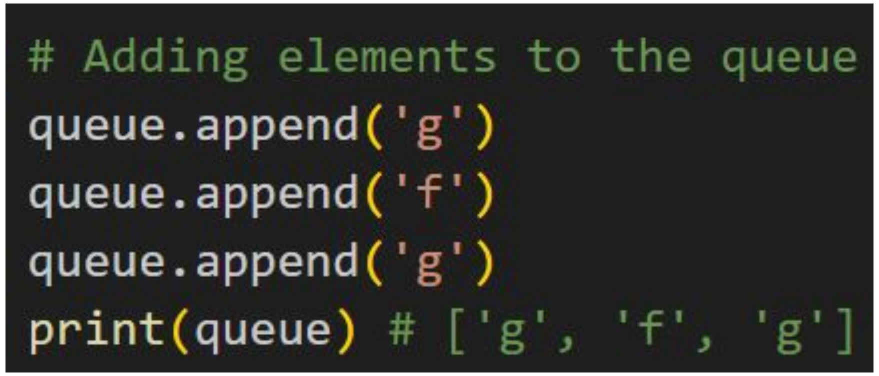
Functions associated with queue

- Enqueue: Adds an item to the queue (Overflow condition)
- Dequeue: Removes an item from the queue (Underflow condition)
- Front: Get the front item from queue
- Rear: Get the last item from queue

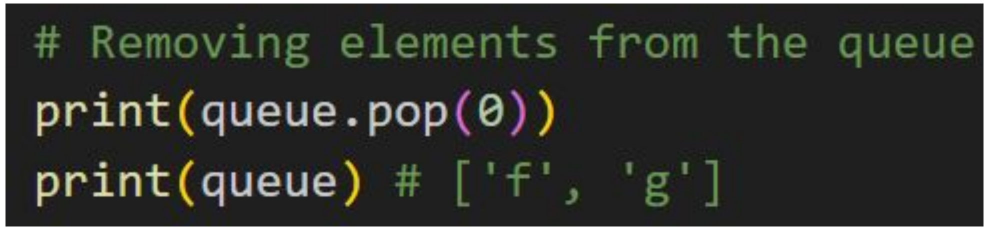
Implementing queue using lists



```python
queue = []
```

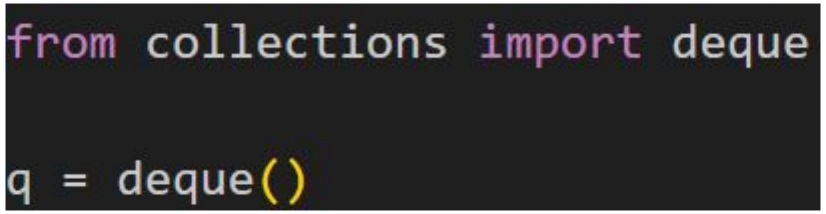


```python
# Adding elements to the queue
queue.append('g')
queue.append('f')
queue.append('g')
print(queue) # ['g', 'f', 'g']
```

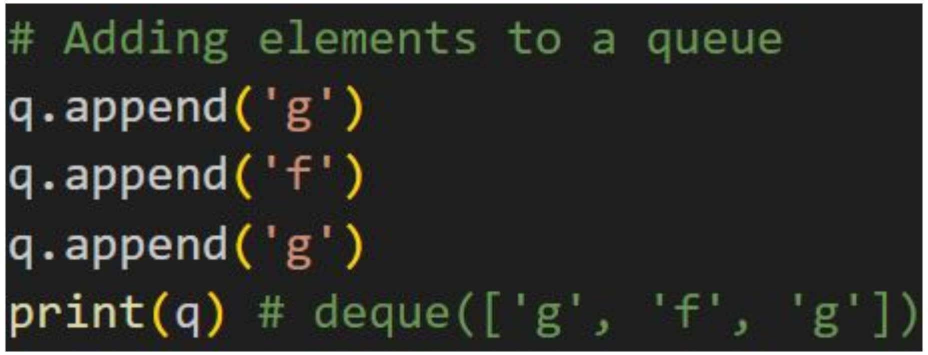


```python
# Removing elements from the queue
print(queue.pop(0))
print(queue) # ['f', 'g']
```

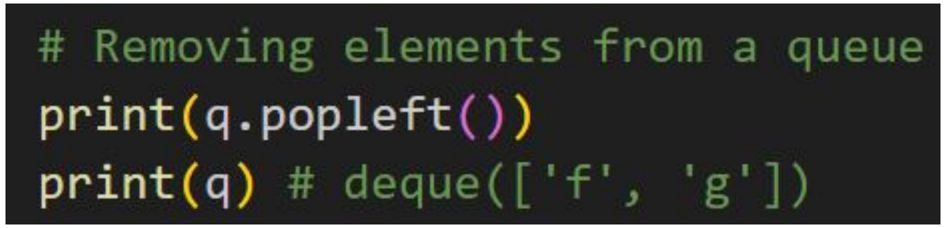
Implementing using dequeue and queue



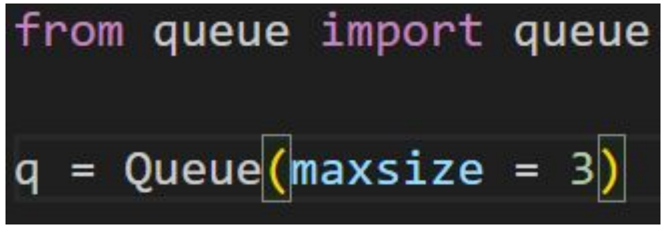
```python
from collections import deque

q = deque()
```



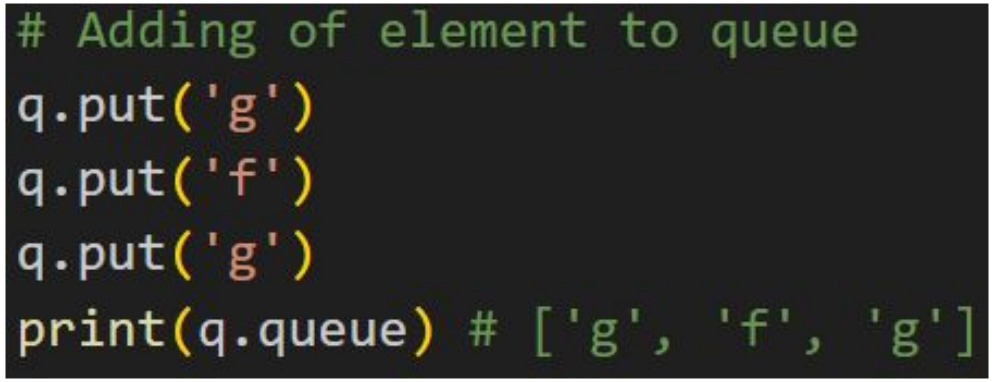
```python
# Adding elements to a queue
q.append('g')
q.append('f')
q.append('g')
print(q) # deque(['g', 'f', 'g'])
```



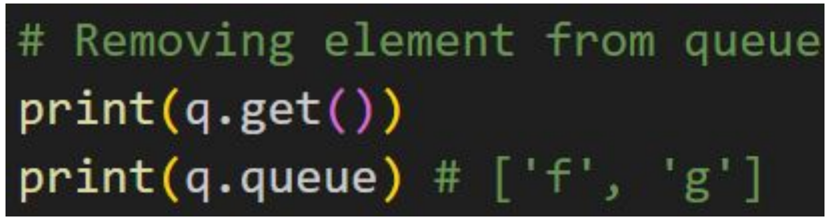
```python
 # Removing elements from a queue
 print(q.popleft())
 print(q) # deque(['f', 'g'])
```



```python
from queue import queue

q = Queue(maxsize = 3)
```

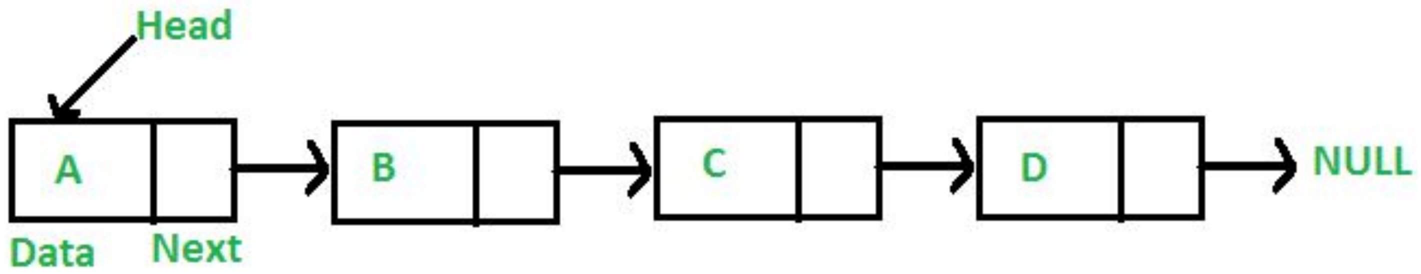


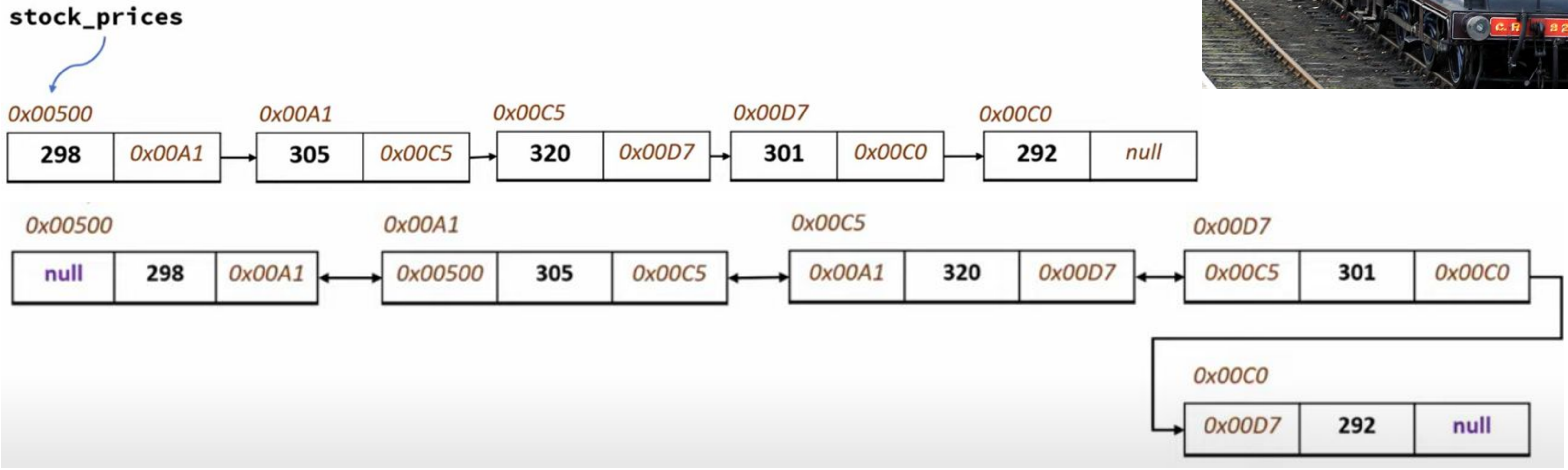
```python
# Adding of element to queue
q.put('g')
q.put('f')
q.put('g')
print(q.queue) # ['g', 'f', 'g']
```



```python
# Removing element from queue
print(q.get())
print(q.queue) # ['f', 'g']
```

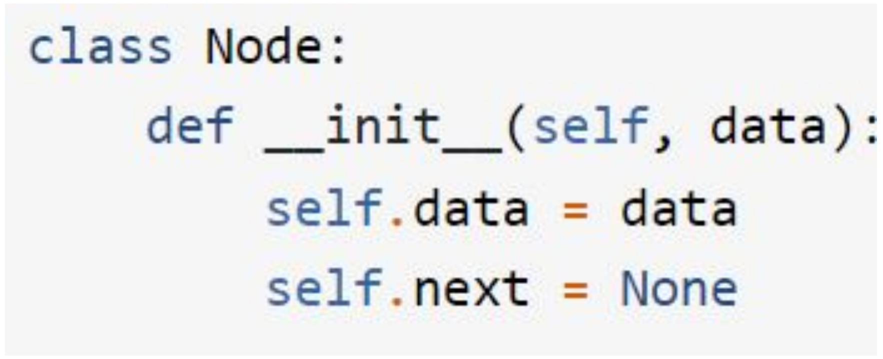
Linked List


Head
A
B
C
D
NULL
Data
Next

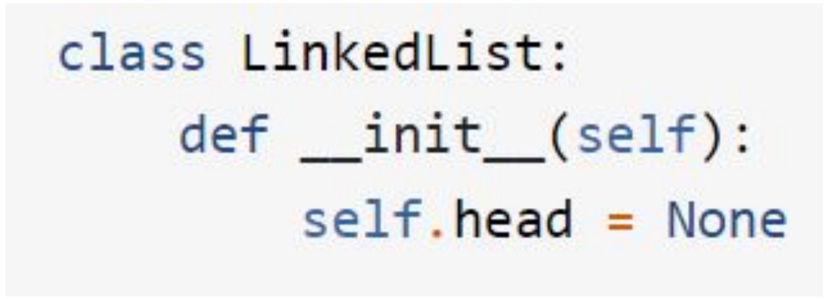

stock_prices
0x00500
298
0x00A1
0x00A1
305
0x00C5
0x00C5
320
0x00D7
0x00D7
301
0x00C0
0x00C0
292
null
0x00500
null
298
0x00A1
0x00A1
0x00500
305
0x00C5
0x00C5
0x00A1
320
0x00D7
0x00D7
0x00C5
301
0x00C0
0x00C0
0x00D7
292
null

Methods associated with linked lists

- insert() : Add an item to the linked list at the head of the list
- find() : Find an item within the linked list
- Remove() : Remove a given item with a given value
- is_empty() : Returns whether the linked list is empty or not
- get_count() : Returns the number of items in the linked list



```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```



```
class LinkedList:
    def __init__(self):
        self.head = None
```

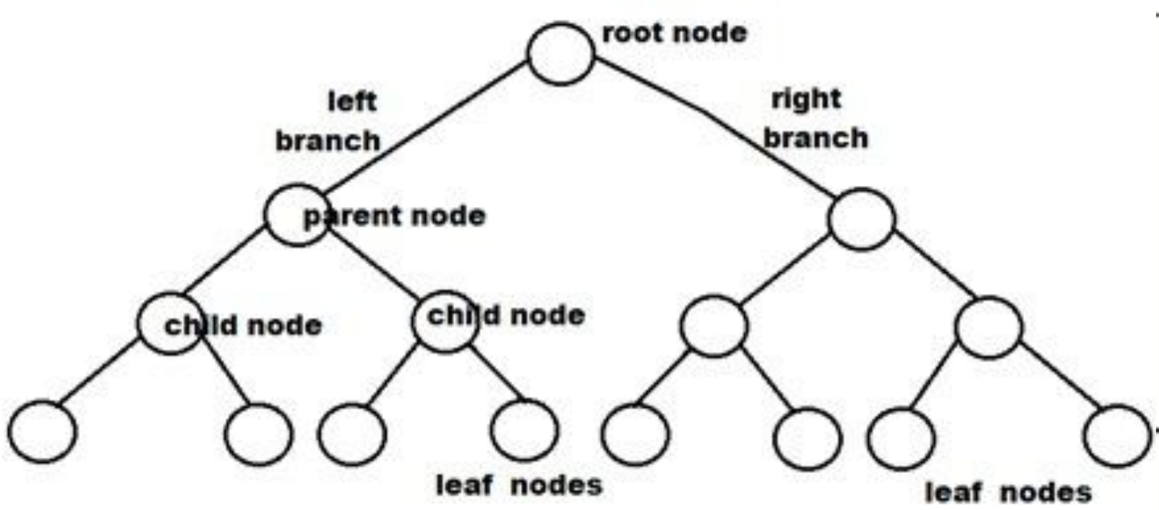
Cont…

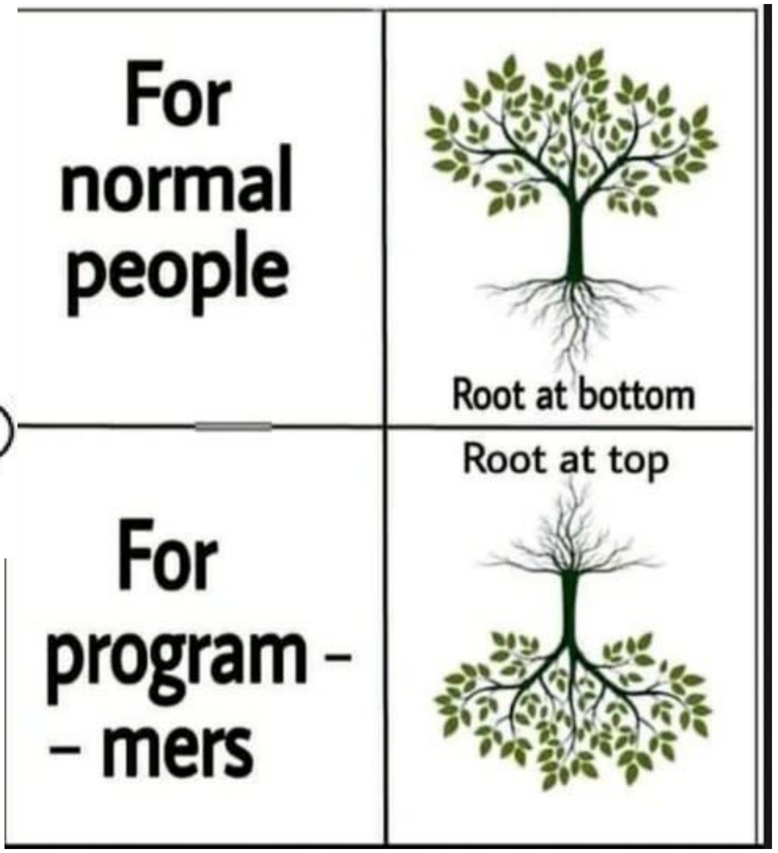
Binary tree

Data

Pointer to left child

Pointer to the right child





Tree Traversal

Depth First Traversals

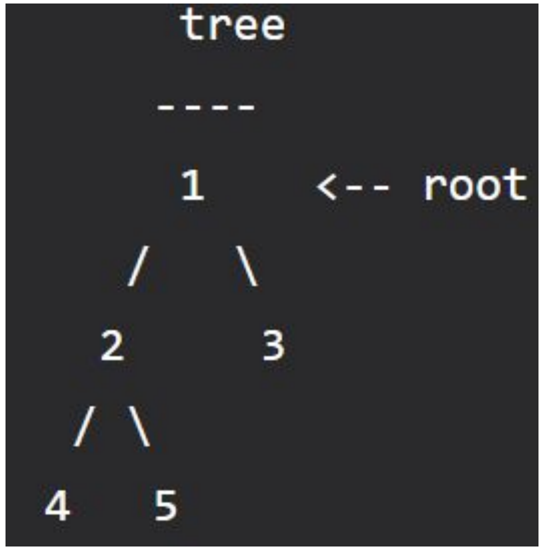
Inorder (Left, Root, Right) : 4 2 5 1 3

Preorder (Root, Left, Right) : 1 2 4 5 3

Postorder (Left, Right, Root) : 4 5 2 3 1

Breadth-First or Level Order Traversal

1 2 3 4 5



```
          tree
         ----
          1     <-- root
        /   \
       2      3
     / \
    4   5
```

Standard Package

- Built-in library

random

math

array

datetime

Third party language packages

Cont…

```
pip install <package_name>
pip install numpy
```

→

```
import package_name
import numpy
```

numpy

pygame

pandas

matplotlib

Reference

Lemonaki, D. (2023h, February 21). Python array tutorial – define, index, methods. freeCodeCamp.org. https://www.freecodecamp.org/news/python-array-tutorial-define-index-methods/

OluseyeJeremiah. (2023, April 6). Multi-dimensional arrays in python – matrices explained with examples. freeCodeCamp.org. https://www.freecodecamp.org/news/multi-dimensional-arrays-in-python/

THANK YOU