In [ ]:

```
#Balanced Parenthesis
Well formed Parenthesis
How many types of Parentesis we have:
    square bracket = []
    curly braces = {}
    brackets = ()
[({})] ==> Balanced Parenthesis
() ==> Balanced
([{}}])  ==> Not Balanced
```

In [ ]:

```
string  = "{([])}"
return True if brackets are balanced
else: return False

Opening Bracket ==> Push in into the Stack
Closing Bracket ==> Pop it from it from the stack

Open brackets must be closed by the same type of brackets.
Open brackets must be closed in the correct order.
Every close bracket has a corresponding open bracket of the same type.
```

In [ ]:

```
#Steps":1 Create a empty stack
Step 2: Iterate over the Given string
step 3: If we are having opening bracket just pushed it into the stack
Step 4: If we are having closing bracked just check stack top. if stack top and closing bracket
    both are in pairs that means they are balanced just poped the opening bracket
    if they both are not in pair just return False
Step: if stack length is 0 that means brackets are balanced else not
```

In [7]:

```
str="[{}]"
stack=[]
for i in range(len(str)):
    if (str[i]=="[" or str[i]=="{" or str[i]=="("):
        stack.append(str[i])
    elif (len(stack)!=0 and stack[-1]=="[" and str[i]=="]"):
        stack.pop()
    elif (len(stack)!=0 and stack[-1]=="{" and str[i]=="}"):
        stack.pop()
    elif (len(stack)!=0 and stack[-1]=="(" and str[i]==")"):
        stack.pop()
    else:
        print(False)
if len(stack)==0:
    print(True)
else:
    print(False)
```

```
True
```

In [ ]:

```
#Tower of Hanoi--> It is mathematical puzzle in which we have 3 rods and N number of disk.
Objective: You need to move entire stack to another rod obeying the following rules:
        Only one disk can move at a time
        Each move consists of taking upper disk from one of the stack it top of another stack
        No disk may be placed on top of smaller disk
```

In [ ]:

```
Applications of Stack:
    1.Balanced Parenthesis
    2.Towr of hanoi
    3.Postfix evaluation
    4.Prefix Evalaution
    5.Back Button
```

In [ ]:

```
Two Sum problem
array=[1,2,3,5]
key = 8
```

In [8]:

```
array=[1,2,3,5]
key = 8
for i in range(len(array)):
    for j in range(i+1,len(array)):
        if array[i]+array[j]==key:
            print([i,j])
```

```
[2, 3]
```

In [14]:

```
#ith node of a linkedlist
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
def printll(head):
    while head is not None:
        print(head.data)
        head=head.next
    print("None")
#Ith node
def ithNode(head,i):
    count=0
    while head is not None:
        if count==i:
            return head.data
        count=count+1
        head=head.next
#Linear Seach
def linear_search(head,key):
    count=0
    while head is not None:
        if head.data==key:
            return count+1
        count=count+1
        head=head.next


c1=Node(10)
c2=Node(20)
c3=Node(30)
c4=Node(40)
c1.next=c2
c2.next=c3
c3.next=c4
printll(c1)
print(ithNode(c1,3))
print(linear_search(c1,40))
```

```
10
20
30
40
None
40
4
```

In [ ]:

```
str-{([])}
push -->3
pop -->3
```

In [8]:

```
x= {1: {'Username': 'saro', 'emailid': 'saor@gamil.com', 'Address': 'saalem', 'password': 'Saro@1301', 'Ph_num': '7878787878'}}
y=x[1]["Username"]
user_name=input("Enter user name")
if user_name==y:
    print("You are alreay exist")
else:
    print("Congratulation you are registerd")
```

```
Enter user namename
Congratulation you are registerd
```

In [ ]: