ANALYSE THE CODE

```
# tower of hanoi

def TowerOfHanoi(n , from_rod, to_rod, aux_rod):
    if n == 1:
            print "Move disk 1 from rod",from_rod,"to rod",to_rod
            return
    TowerOfHanoi(n-1, from_rod, aux_rod, to_rod)
    print "Move disk",n,"from rod",from_rod,"to rod",to_rod
    TowerOfHanoi(n-1, aux_rod, to_rod, from_rod)

# Driver code
n = 4
TowerOfHanoi(n, \'A\', \'C\', \'B\')
# A, C, B are the name of rods
```

```
# function to print the pattern
def pattern(n):

    # traverse through the elements
    # in n assuming it as a string
    for i in n:

            # print | for every line
            print("|", end = "")

            # print i number of * s in
            # each line
            print("*" * int(i))

# get the input as string
n = "41325"
pattern(n)
```

```
# Python3 Program to demonstrate
# staircase pattern

# function definition
def pattern(n):

    # for loop for rows
    for i in range(1,n+1):

            # conditional operator
            k =i + 1 if(i % 2 != 0) else i

            # for loop for printing spaces
            for g in range(k,n):
                    if g>=k:
                            print(end=" ")

            # according to value of k carry
            # out further operation
            for j in range(0,k):
                    if j == k - 1:
                            print(" * ")
                    else:
                            print(" * ", end = " ")


# Driver code
n = 10
pattern(n)
```

```
# A simple example class
class Test:

    # A sample method
    def fun(self):
            print("Hello")

# Driver code
obj = Test()
obj.fun()
```

```
# A Sample class with init method
class Person:

    # init method or constructor
    def __init__(self, name):
        self.name = name

    # Sample Method
    def say_hi(self):
        print('Hello, my name is', self.name)

p = Person('el sanhez')
p.say_hi()


# Python program to show that the variables with a value
# assigned in class declaration, are class variables and
# variables inside methods and constructors are instance
# variables.

# Class for Computer Science Student
class CSStudent:

    # Class Variable
    stream = 'cse'

    # The init method or constructor
    def __init__(self, roll):

        # Instance Variable
        self.roll = roll

# Objects of CSStudent class
a = CSStudent(101)
b = CSStudent(102)

print(a.stream) # prints "cse"
print(b.stream) # prints "cse"
print(a.roll) # prints 101

# Class variables can be accessed using class
# name also
print(CSStudent.stream) # prints "cse"

# Python program to show that we can create
# instance variables inside methods
# Class for Computer Science Student
class CSStudent:

    # Class Variable
    stream = 'cse'

    # The init method or constructor
    def __init__(self, roll):

            # Instance Variable
            self.roll = roll

    # Adds an instance variable
    def setAddress(self, address):
            self.address = address

    # Retrieves instance variable
    def getAddress(self):
            return self.address

# Driver Code
a = CSStudent(101)
a.setAddress("On the earth")
print(a.getAddress())

class MyClass:

    # Hidden member of MyClass
    __hiddenVariable = 0

    # A member method that changes
    # __hiddenVariable
    def add(self, increment):
            self.__hiddenVariable += increment
            print (self.__hiddenVariable)

# Driver code
myObject = MyClass()
myObject.add(2)
myObject.add(5)

# This line causes error
print (myObject.__hiddenVariable)
```

```
# A Python program to demonstrate that hidden
# members can be accessed outside a class
class MyClass:

    # Hidden member of MyClass
    __hiddenVariable = 10

# Driver code
myObject = MyClass()
print(myObject._MyClass__hiddenVariable)

class Test:
    def __init__(self, a, b):
            self.a = a
            self.b = b

    def __repr__(self):
            return "Test a:%s b:%s" % (self.a, self.b)

    def __str__(self):
            return "From str method of Test: a is %s," \
                        "b is %s" % (self.a, self.b)

# Driver Code
t = Test(1234, 5678)
print(t) # This calls __str__()
print([t]) # This calls __repr__()

class Test:
    def __init__(self, a, b):
            self.a = a
            self.b = b

    def __repr__(self):
            return "Test a:%s b:%s" % (self.a, self.b)

# Driver Code
t = Test(1234, 5678)
print(t)
```