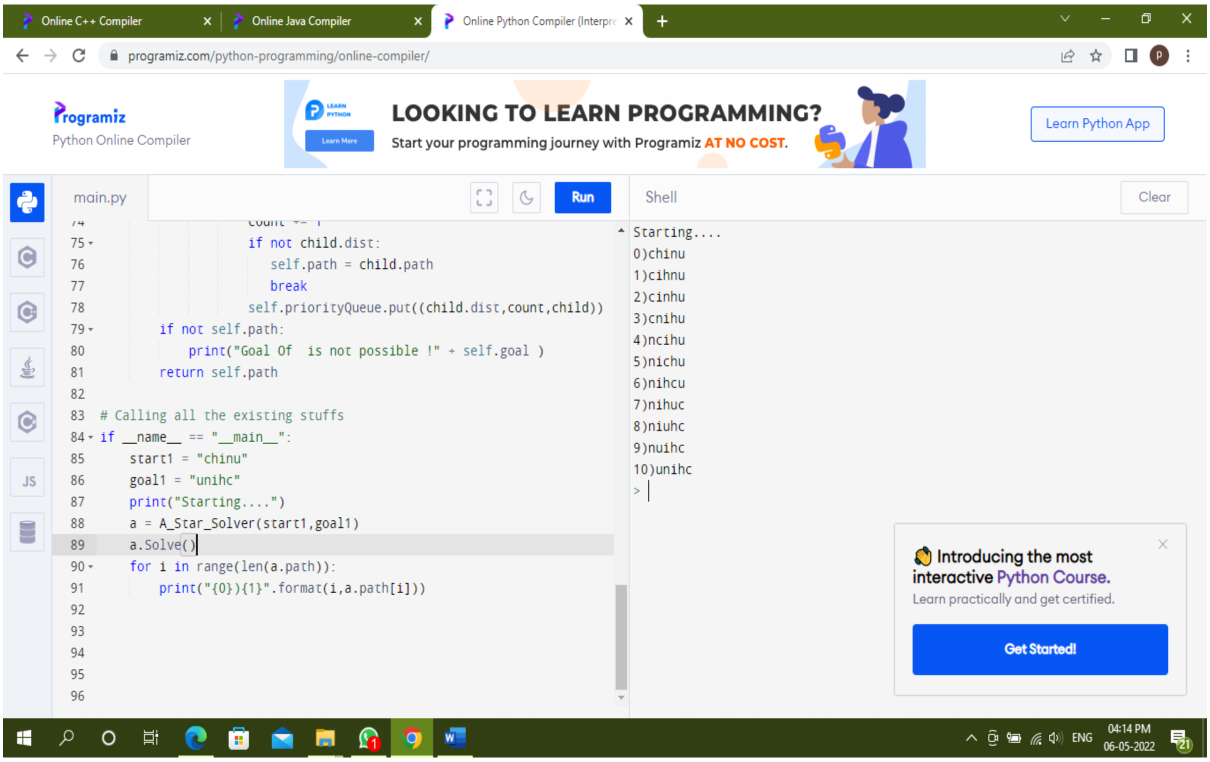
ASSIGNMENT:2

```
#Ass-2
#Lp-2
from queue import PriorityQueue

#Creating Base Class
class State(object):
    def __init__(self, value, parent, start = 0, goal = 0):
        self.children = []
        self.parent = parent
        self.value = value
        self.dist = 0
        if parent:
            self.start = parent.start
            self.goal = parent.goal
            self.path = parent.path[:]
            self.path.append(value)

        else:
            self.path = [value]
            self.start = start
            self.goal = goal

    def GetDistance(self):
        pass
    def CreateChildren(self):
        pass
```

```
# Creating subclass
class State_String(State):
  def __init__(self, value, parent, start = 0, goal = 0 ):
    super(State_String, self).__init__(value, parent, start, goal)
    self.dist = self.GetDistance()

  def GetDistance(self):
      if self.value == self.goal:
        return 0
      dist = 0
      for i in range(len(self.goal)):
        letter = self.goal[i]
        dist += abs(i - self.value.index(letter))
      return dist

  def CreateChildren(self):
      if not self.children:
        for i in range(len(self.goal)-1):
          val = self.value
          val = val[:i] + val[i+1] + val[i] + val[i+2:]
          child = State_String(val, self)
          self.children.append(child)

# Creating a class that hold the final magic
class A_Star_Solver:
  def __init__(self, start, goal):
    self.path = []
    self.vistedQueue =[]
    self.priorityQueue = PriorityQueue()
    self.start = start
    self.goal = goal
    def Solve(self):
        startState = State_String(self.start,0,self.start,self.goal)


        count = 0
        self.priorityQueue.put((0,count, startState))
        while(not self.path and self.priorityQueue.qsize()):
              closesetChild = self.priorityQueue.get()[2]
              closesetChild.CreateChildren()
              self.vistedQueue.append(closesetChild.value)
              for child in closesetChild.children:
                  if child.value not in self.vistedQueue:
                   count += 1
                   if not child.dist:
                      self.path = child.path
                      break
                   self.priorityQueue.put((child.dist,count,child))
        if not self.path:
            print("Goal Of  is not possible !" + self.goal )
        return self.path


# Calling all the existing stuffs
if __name__ == "__main__":
    start1 = "chinu"
    goal1 = "unihc"
    print("Starting....")
    a = A_Star_Solver(start1,goal1)
    a.Solve()
    for i in range(len(a.path)):
        print("{0}){1}".format(i,a.path[i]))
```



```python
                count += 1
                if not child.dist:
                    self.path = child.path
                    break
                self.priorityQueue.put((child.dist,count,child))
        if not self.path:
            print("Goal Of  is not possible !" + self.goal )
        return self.path

# Calling all the existing stuffs
if __name__ == "__main__":
    start1 = "chinu"
    goal1 = "unihc"
    print("Starting....")
    a = A_Star_Solver(start1,goal1)
    a.Solve()
    for i in range(len(a.path)):
        print("{0}){1}".format(i,a.path[i]))
```

```
Starting....
0)chinu
1)cihnu
2)cinhu
3)cnihu
4)ncihu
5)nichu
6)nihcu
7)nihuc
8)niuhc
9)nuihc
10)unihc
>
```