**1) Define a recursive procedure in Python and in Prolog to find the sum of 1st n terms of an equal-interval series given the 1st term and the interval.**

**Ans:**

**Python:**

def seriessum(number,interval,first):

if(number==0):

return 0

elif (number>=1):

return seriessum(number-1, interval, first)+first+(number-1)\*interval

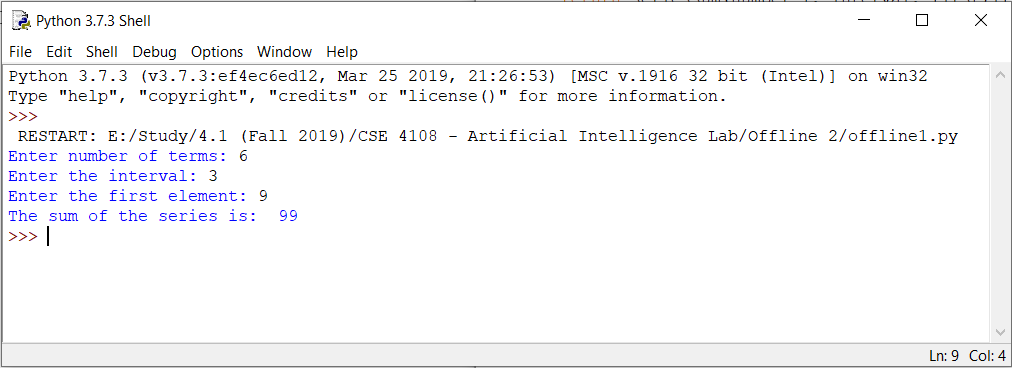
#Main

number=int(input('Enter number of terms: '))

interval=int(input('Enter the interval: '))

first=int(input('Enter the first element: '))

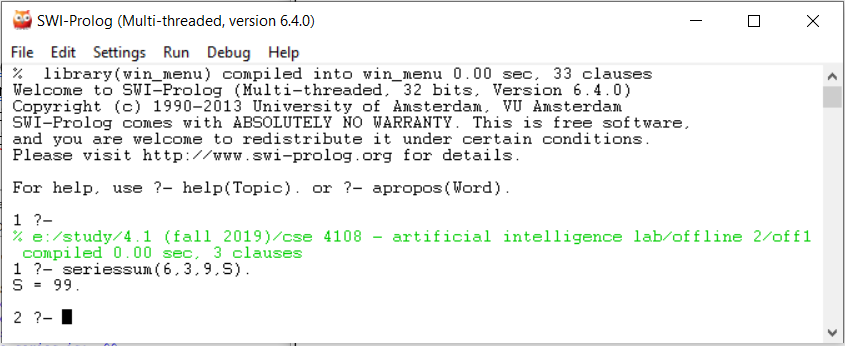
result=seriessum(number,interval,first)

print('The sum of the series is: ',result)

**Prolog:**

seriessum(0,\_,\_,0):- !.

seriessum(N,I,F,S2):- N1 is N-1, seriessum(N1,I,F,S1), S2 is S1+F+(N-1)\*I.



**2) Define a recursive procedure in Python and in Prolog to find the length of a path between two vertices of a directed weighted graph.**

**Prolog:**

neighbor(i,a,35). neighbor(i,b,45). neighbor(a,c,20).

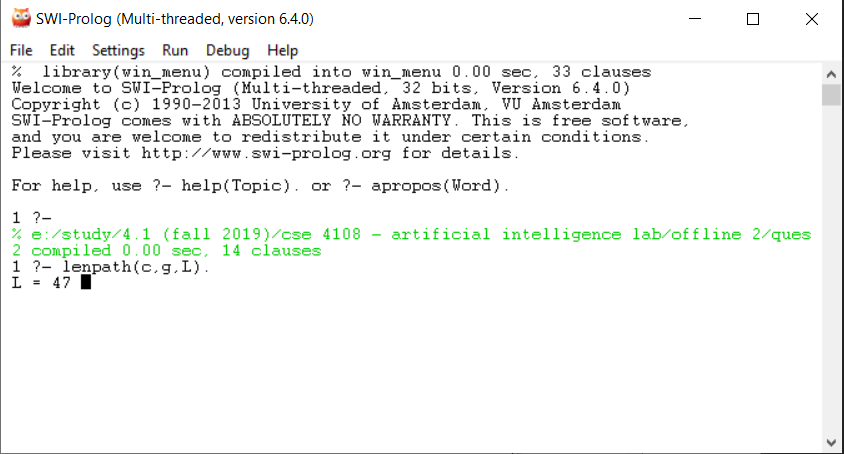
neighbor(a,d,30). neighbor(b,d,25). neighbor(b,e,35).

neighbor(b,f,27). neighbor(c,d,30). neighbor(c,g,47).

neighbor(d,g,30). neighbor(e,g,25).

lenpath(X,Y,L):-neighbor(X,Y,L).

lenpath(X,Y,L):-neighbor(X,Z,L1),lenpath(Z,Y,L2),L is L1+L2.



**3) Modify the Python and Prolog codes demonstrated above to find h2 and h3 discussed above.**

**h2:**

**Python:**

in\_state=[(1,1,2),

(2,1,3),

(3,2,1),

(8,1,1),

(4,2,3),

(7,3,2),

(6,2,2),

(5,3,3)]

goal\_state=[(1,1,1),

(2,1,2),

(3,1,3),

(8,2,1),

(4,2,3),

(7,3,1),

(6,3,2),

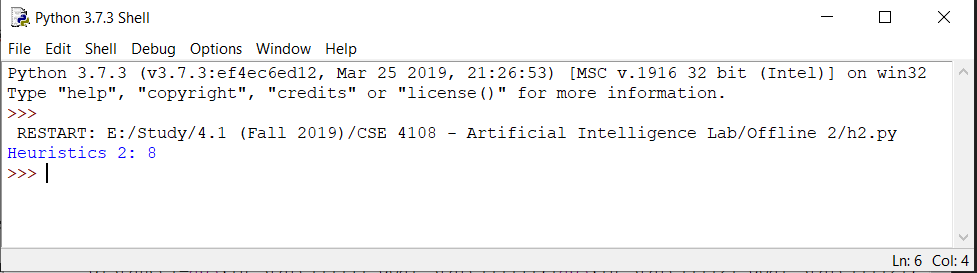
(5,3,3)]

distance=0

for i in range(8):

distance+=abs(in\_state[i][1]-goal\_state[i][1])+abs(in\_state[i][2]-goal\_state[i][2])

print('Heuristics 2:',distance)



**Prolog:**

gtp(1,1,1). gtp(2,1,2). gtp(3,1,3). gtp(4,2,3). gtp(5,3,3). gtp(6,3,2). gtp(7,3,1). gtp(8,2,1). gblnk(2,2).

tp(1,1,2). tp(2,1,3). tp(3,2,1). tp(4,2,3). tp(5,3,3). tp(6,2,2). tp(7,3,2). tp(8,1,1). blnk(3,1).

go:- calcH(1,[],L), sumList(L,V),write('Heuristics: '),write(V).

calcH(9,X,X):-!. calcH(T,X,Y):- dist(T,D), append(X,[D],X1), T1 is T+1, calcH(T1,X1,Y).

dist(T,V):-tp(T,A,B), gtp(T,C,D), V is abs(A-C) + abs(B-D).

sumList([],0):-!. sumList(L,V):-L=[H|T], sumList(T,V1), V is V1+H.

