

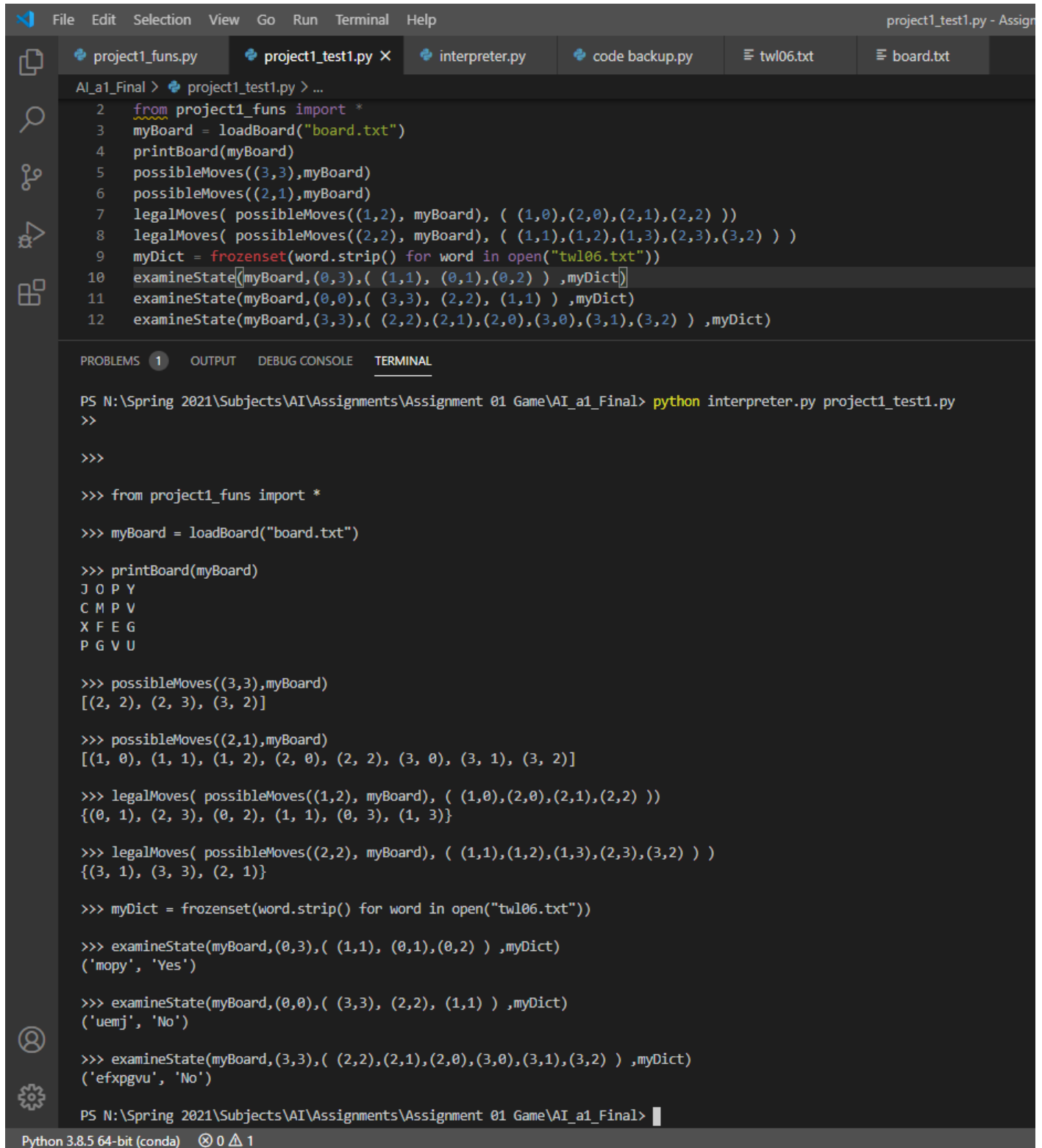
Name: Saurabh Jawahar Kakade

Course: CS570/Spring 2021/Advanced Intelligence Systems

Assignment Title: Fred Flintstone Problem Solving / Program 01 / Part 01

Date: 01/29/2021

Output:



The screenshot shows a Python IDE with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar indicates the active file is 'project1_test1.py - Assign...'. The editor has several tabs: 'project1_funs.py', 'project1_test1.py' (active), 'interpreter.py', 'code backup.py', 'twl06.txt', and 'board.txt'. The code in 'project1_test1.py' is as follows:

```
AI_a1_Final > project1_test1.py > ...
2  from project1_funs import *
3  myBoard = loadBoard("board.txt")
4  printBoard(myBoard)
5  possibleMoves((3,3),myBoard)
6  possibleMoves((2,1),myBoard)
7  legalMoves( possibleMoves((1,2), myBoard), ( (1,0),(2,0),(2,1),(2,2) ))
8  legalMoves( possibleMoves((2,2), myBoard), ( (1,1),(1,2),(1,3),(2,3),(3,2) ) )
9  myDict = frozenset(word.strip() for word in open("twl06.txt"))
10 examineState(myBoard,(0,3),( (1,1), (0,1),(0,2) ),myDict)
11 examineState(myBoard,(0,0),( (3,3), (2,2), (1,1) ),myDict)
12 examineState(myBoard,(3,3),( (2,2),(2,1),(2,0),(3,0),(3,1),(3,2) ),myDict)
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS N:\Spring 2021\Subjects\AI\Assignments\Assignment 01 Game\AI_a1_Final> python interpreter.py project1_test1.py
>>

>>>

>>> from project1_funs import *

>>> myBoard = loadBoard("board.txt")

>>> printBoard(myBoard)
J O P Y
C M P V
X F E G
P G V U

>>> possibleMoves((3,3),myBoard)
[(2, 2), (2, 3), (3, 2)]

>>> possibleMoves((2,1),myBoard)
[(1, 0), (1, 1), (1, 2), (2, 0), (2, 2), (3, 0), (3, 1), (3, 2)]

>>> legalMoves( possibleMoves((1,2), myBoard), ( (1,0),(2,0),(2,1),(2,2) ))
{(0, 1), (2, 3), (0, 2), (1, 1), (0, 3), (1, 3)}

>>> legalMoves( possibleMoves((2,2), myBoard), ( (1,1),(1,2),(1,3),(2,3),(3,2) ) )
{(3, 1), (3, 3), (2, 1)}

>>> myDict = frozenset(word.strip() for word in open("twl06.txt"))


>>> examineState(myBoard,(0,3),( (1,1), (0,1),(0,2) ),myDict)
('mopy', 'Yes')

>>> examineState(myBoard,(0,0),( (3,3), (2,2), (1,1) ),myDict)
('uemj', 'No')

>>> examineState(myBoard,(3,3),( (2,2),(2,1),(2,0),(3,0),(3,1),(3,2) ),myDict)
('efxpgvu', 'No')



PS N:\Spring 2021\Subjects\AI\Assignments\Assignment 01 Game\AI_a1_Final>
```

The status bar at the bottom indicates 'Python 3.8.5 64-bit (conda)' and shows a memory usage of '0 0 1'.

AI_a1_Final >  project1_funs.py > ...

```
1  #AIS Programming assignment: 01 (Part 01)
2  #Programmer: Saurabh Jawahar Kakade
3  #NAU Email ID: sk2354@nau.edu
4
5  import string
6
7  #load board function
8  def loadBoard(newboard):
9      board = open(newboard,"r")
10     new_board = []
11     for i in board:
12         new_board.append(i.split())
13     return new_board
14
15 #print board function
16 def printBoard(newboard):
17     for i in newboard:
18         print(' '.join(map(str,i)))
19
20 #possible move function
21 def possibleMoves(moves,newboard):
22     newpmoves = list() #possible moves list
23     x = moves[0] #row
24     y = moves[1] #column
25
26     l = len(newboard)
27
28     #top coordinates
29     if (x-(x-1)==1) & (y-(y-1)==1):
30         if (l>(x-1)>-1) & (l>(y-1)>-1):
31             newpmoves = newpmoves + [(x-1,y-1)]
32
33     if (x-(x-1)==1):
34         if (l>(x-1)>-1) & (l>(y)>-1):
35             newpmoves = newpmoves + [(x-1,y)]
36
37     if (x-(x-1)==1) & (((y+1)-y)==1):
38         if (l>(x-1)>-1) & (l>(y+1)>-1):
```

```
36
37     if (x-(x-1)==1) & (((y+1)-y)==1):
38         if (1>(x-1)>-1) & (1>(y+1)>-1):
39             newpmoves = newpmoves + [(x-1,y+1)]
40
41     #left coordinates
42     if (y-(y-1)==1):
43         if (1>(x)>-1) & (1>(y-1)>-1):
44             newpmoves = newpmoves + [(x,y-1)]
45
46     #right coordinates
47     if (((y+1)-y)==1):
48         if (1>(x)>-1) & (1>(y+1)>-1):
49             newpmoves = newpmoves + [(x,y+1)]
50
51     #bottom coordinates
52     if ((x+1)-x==1) & (y-(y-1)==1):
53         if (1>(x+1)>-1) & (1>(y-1)>-1):
54             newpmoves = newpmoves + [(x+1,y-1)]
55
56     if ((x+1)-x==1):
57         if (1>(x+1)>-1) & (1>(y)>-1):
58             newpmoves = newpmoves + [(x+1,y)]
59
60     if ((x+1)-x==1) & (((y+1)-y)==1):
61         if (1>(x+1)>-1) & (1>(y+1)>-1):
62             newpmoves = newpmoves + [(x+1,y+1)]
63
64     else:
65         print("error")
66
67     return newpmoves
68
69 #Legal moves function
70 def legalMoves(m0,m1): #m0 is all possible moves and m1 is visited moves
71     l1 = set(m1)
72     l2 = set(m0)
73     lm = set(l2 - l1) #legal moves set
74     # print(lm)
75     return lm
```

Al_a1_Final >  project1_funs.py >  possibleMoves

```
72     l2 = set(m0)
73     lm = set(l2 - l1) #legal moves set
74     # print(lm)
75     return lm
76
77 #examine state function
78 def examineState(myboard,m1,m2,myDict):
79     #m1=current position
80     #m2=current path
81     es = list() #examine state list variable
82
83
84     for i in m2:
85         x, y = i      #x=row and y=column
86         es.append(myboard[x][y])
87
88     x, y = m1
89
90     es.append(myboard[x][y])
91
92     #fes variable = final examine state
93     fes = "".join(map(str,es))
94
95
96     if fes.lower() in myDict:
97         print((fes.lower(),"Yes"))
98     else:
99         print((fes.lower(),"No"))
100
101
102     #end of code
103
```