Faculty of Engineering, University Of Jaffna

Department Of Computer Engineering

EC9640 – Artificial Intelligence

Lab 02

Date: 5th June 2022 Duration: 3 Hours

Instructions:

- Any plagiarized work will be given 0 marks.
- Submit your lab work according to the instructions given below **on/before given deadline** via teams.
- Failure to adhere to any of the above instructions may also result in zero marks.

Introduction to Python (ref: www.kaggle.com) [10 minutes - Reading]

You may try in out the commands in Jupiter notebook (Start>Anaconda>Anaconda Navigator>Jupiter Notebook)

Note intent decides whether you are in a class, definition, loop, or if condition thus if indent thus the code should ALWAYS be formatted correctly.

Comments

the symbol '#' is used for inline comments in python Print (similar to 'disp' in MATLAB).

print(5 / 2) # prints the value of 5/2

Numbers and arithmetic in Python

Operator	Name	Description
`a + b`	Addition	Sum of a and b
`a - b`	Subtraction	Difference of a and b
`a * b`	Multiplication	Product of a and b
`a / b`	True division	Quotient of a and b
`a // b`	Floor division	Quotient of a and b, removing fractional parts

a % b	Modulus	Integer remainder after division of a by b
a ** b	Exponentiation	a raised to the power of b
-a	Negation	The negative of a

Built-in functions for working with numbers

min(1, 2, 3) #gives minimum of all inputs

max(1, 2, 3) #gives maximum of all inputs
abs(32) # gives absolute value of input

Getting Help

help(round) #here round is the name of a function

Functions (this is analogous to functions in MATLAB)

```
def least_difference(a, b, c): # def function_name(input_1, input_2, … ,
input_n)            diff1 = abs(a - b)

diff2 = abs(b - c)

 diff3 = abs(a - c)

 return min(diff1, diff2, diff3)
```

Lists

primes = [2, 3, 5, 7] # list of numbers

planets = ['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune'] # list of strings my_favourite_things = [32, 'raindrops on roses', help] # list can contain different items
hands = [['J', 'Q', 'K'], ['2', '2', '2'], ['6', 'A', 'K']] # list of lists

Indexing

planets[0] # get zeroth term

planets[-1] # last term

Slicing

planets[0:3] # extract elements zero to three

planets[:3] # again first 3

planets[3:] # 3rd to last

planets[1:-1] # All except the first and last

planets[-3:] # The last 3 elements

planets[3] = 'Malacandra' # change the last element
planets[:3] = ['Mur', 'Vee', 'Ur'] # change 3 elements at
once len(planets) # length of planets

sorted(planets) # sort in alphabetical order

sum(primes) # sum of all elements

max(primes) # maximum of elements

planets.append('Pluto') # add element to end of list

planets.pop() # remove from end of list

planets.index('Earth') # the index location of 'Earth'

"Earth" in planets # returns Boolean output on whether "Earth" is in the list planets

Tuples

t = (1, 2, 3) # initializing a
tuple

t = 1, 2, 3 # equivalent to
above

t[0] = 100 is not valid as values in a tuple cannot be
modified

a, b = b, a # swapping variables a and b

Loops

Eg.1:

```
planets = ['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus',
'Neptune'] for planet in planets:
    print(planet, end=' ') # print all on same line
```

Out:
Mercury Venus Earth Mars Jupiter Saturn Uranus Neptune

Eg. 2

```
product = 1
for mult in
multiplicands:
product = product * mult
product
```

Out:
360

Eg. 3

```
s = 'steganograpHy is the practicE of conceaLing a file, message, image, or video
within ano ther fiLe, message, image, Or video.'
msg = ''
# print all the uppercase letters in s, one at a
time for char in s:
    if char.isupper():
       print(char, end='')
```

Out:

HELLO

Eg. 4

```
for i in range(5):
    print("Doing important work. i
=", i)
```

Out:

```
Doing important work. i = 0
Doing important work. i = 1
Doing important work. i = 2
Doing important work. i = 3
Doing important work. i = 4
```

Eg. 5

```
  i = 0
  while i < 10:
  print(i, end=' ')
  i += 1
```

Out:

```
0 1 2 3 4 5 6 7 8 9
```

Class

A class in python is an object with properties and methods. A constructor is a special method in the class which runs immediately as the class is created

Analogy to MATLAB (MATLAB also has classes but this is just for simpler understanding):

-Class: an instance of MATLAB with its own work space

-properties: variables in the work space assigned different values

-methods: functions

-constructor: a start-up script you run to load some initial values into workspace variables

Constructor

Eg.

```
class transportationProblem

def __init__(self, N):

    self.N =N+8
```

Here,
When you run a command like,
Prob = transportationProblem(N=10)

You get,
-An instance of transportationProblem stored in Prob
-With the class property N set to 10
-self refers to that instance of class (as in C++ or Java)

Introduction to Prolog [0 minutes - Reading only if needed]

Installation

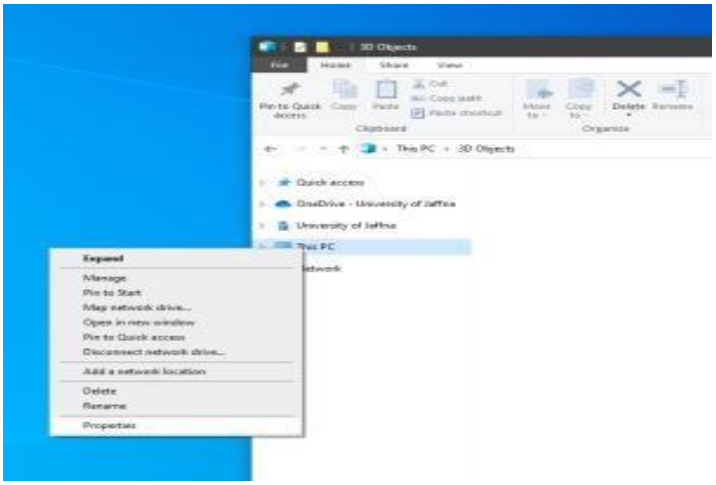
We will be using SWI-Prolog to code in prolog. Prolog can be downloaded from the link below.
https://www.swi-prolog.org/Download.html

Next as prolog is to be interfaced with python I additional library is required. It is the PySwip library. Instructions to install can be found in the link below.

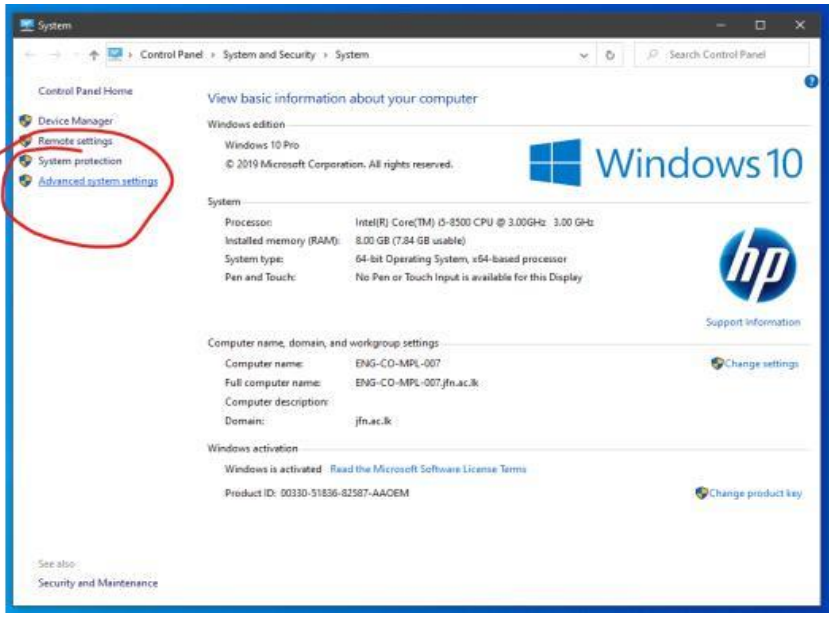
https://github.com/yuce/pyswip/blob/master/INSTALL.md

Windows installation is as follows.

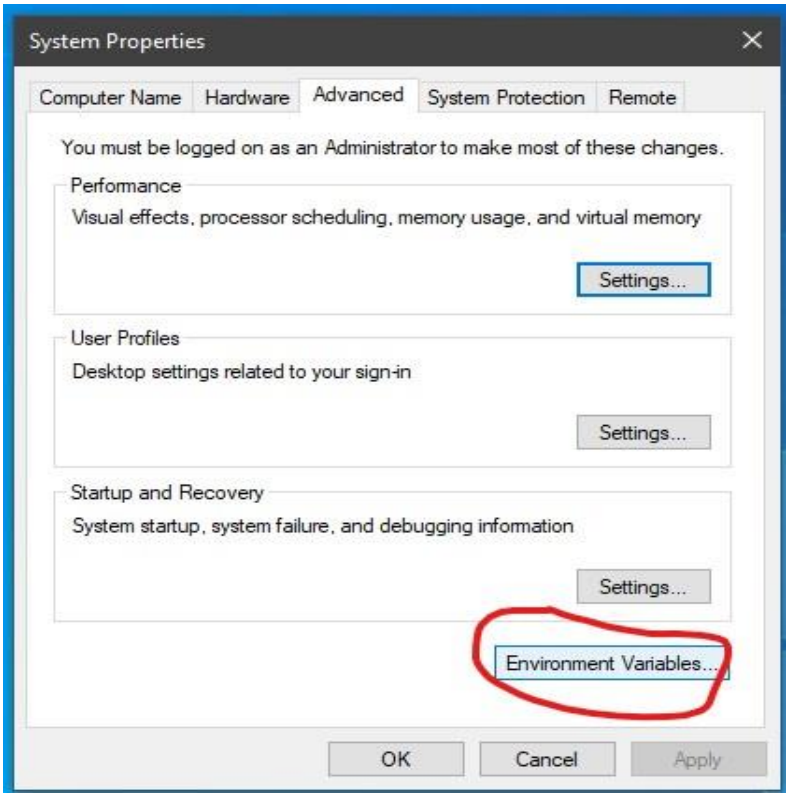
1. Make sure swipl executable is on the PATH.
    a) Go to the path where SWI-Prolog is installed (right click shortcut for SWI-Prolog icon and go to folder location, repeat until you get to the actual folder location). The default path is "C:\Program Files\swipl\bin".
    b) Then go to file explorer. Right click this PC>Properties



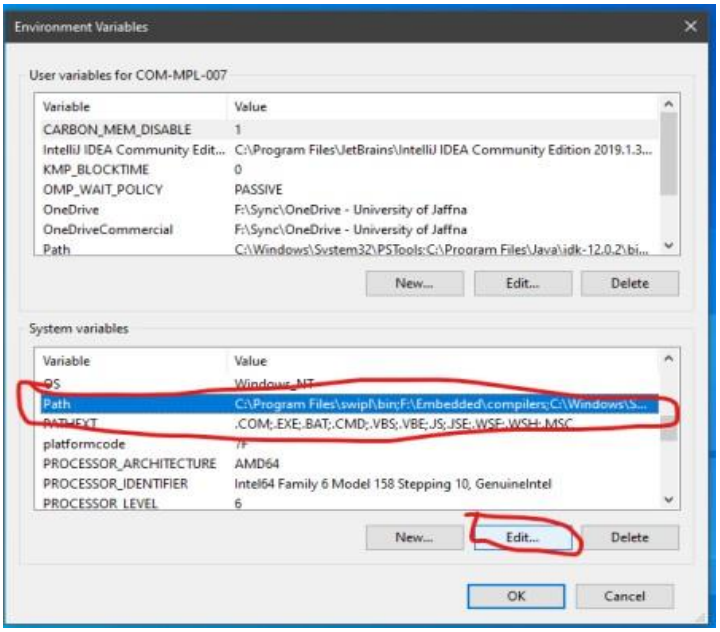
c) In the newly opened properties window. Click Advanced System Settings



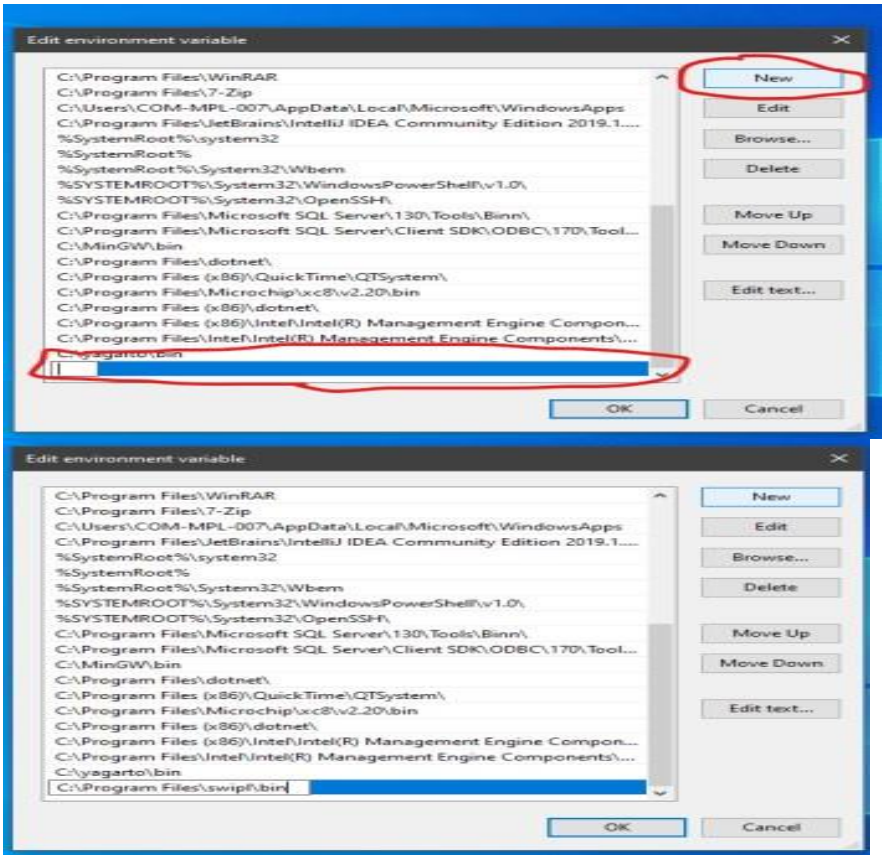
d) In the new window select Environment Variables.



e) In the new window. Select Path then click edit.



f) Next in the Edit Environment Variable window. Press New then type/paste the path you found in a



g) Press "OK" to confirm changes in all required windows.

2. pip install pyswip

3. Run a quick test by running following code at your Python console:

```
from pyswip import Prolog prolog = Prolog()
prolog.assertz("father(michael,john)")
```

Examples in Prolog [40 minutes – Practicing and Understanding]

1 ?- X is 3+2. X = 5.
2 ?- 3+2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated
3 ?- X = 3+2. X = 3+2.
4 ?- 3+2 = X.
X = 3+2.
5 ?- X is +(3,2).
X = 5. 6 ?- 5
is 3+2.
true. 7 ?-
3+2 is 5.
false. 8 ?- X
is 3*2. X = 6
9 ?- X is 3-2.
X = 1.
10 ?- X is -(2,3). X = -1.
11 ?- X is 5-3-1.
X = 1.
12 ?- X is -(5,3,1).
ERROR: is/2: Arithmetic: `(-)/3' is not a function 13
?- X is -(-(5,3),1).
X = 1.
14 ?- X is 3/5.
X = 0.6.
15 ?- X is 3 mod 5.
X = 3.
16 ?- X is 5 mod 3.
X = 2.
17 ?- X is 5^3.
X = 125.
18 ?- X is (5^3)^2.
X = 15625.
19 ?- X = (5^3)^2.
X = (5^3)^2. 20
?- 25 is 5^2.
true.
21 ?- Y is 3+2*4-1.
Y = 10.
22 ?- Y is (3+2)*(4)-(1).
Y = 19.
23 ?- Y is -(*(+(3,2),4),1).
Y = 19.
24 ?- X is 3*2, Y is X*2.

```
X = 6,
Y = 12
25 ?- [X|Y] = [A | [b, c]] .  X
= A,
Y = [b, c].
26 ?- [A | [b, c]] = [X|Y]. A
= X,
Y = [b, c].
27 ?- [B, [p, n, c], 4] = [X | Y].
B = X,
Y = [[p, n, c], 4].
28 ?- [likes(john, mary), X , 1, 2] = [Head | Tail].
Head = likes(john, mary), Tail
= [X, 1, 2].
29 ?- [[1, 2], a, b ] = [Head | Tail]. Head = [1, 2],
Tail = [a, b].
30 ?-  [john] = [Head | Tail].
Head = john,
Tail = [].
31 ?- append([a,b],[c,d],Z).
Z = [a, b, c, d].
32  ?- append(X,[c,d],[x, y, z, c, d]). X = [x, y,
    z] ; false.
33  ?- append(X, Y, [a, b]).
X = [],
Y = [a, b] ;
X = [a],
Y = [b] ;
X = [a, b], Y
= [] ; false.
```

For the following you have to use the attached files "foodMeal.pl", "studentTeacher.pl" and "power.pl".

```
34 ?- food(pizza).
true.
35 ?-  meal(X), lunch(X). X = sandwich ;
   false.
36 ?-  dinner(sandwich). false.
37 ?- meal(X), dinner(X).
X = pizza.
38 ?- meal(What).
What = burger ;
What = sandwich ; What
= pizza.
39 ?- meal(X), dinner(Y).
X = burger,
Y = pizza ;
X = sandwich,
Y = pizza ;
X = Y, Y = pizza.
40 ?- studies(charlie, What).
What = csc135.
41 ?- professor(kirke,
   Students). Students =
   charlie ; Students = olivia.
42 ?- studies(Who, csc135).
   Who = charlie ; Who =
   olivia.
```

43 ?- power(2,2,X). X = 4 ;
false.

Simple exercises in Prolog [60 minutes – Exercises]

Task 1:

Write a function/rules to find the factorial of any given number.
Example Output:
1?- fact(-5,Y).
false.
2 ?fact(5,Y).
Y = 120 ;
false.
3 ?-fact(0,Y).
Y = 1 ;
false.
4 ?- fact(1,Y).
Y = 1 ;
false.
5 ?- fact(1.5,Y).
ERROR: mod/2: Type error: `integer' expected, found `1.5' 6
?- fact(4,24).
true ;
false.

Task 2:

Write a program to find if a particular element is in a given list of elements.
Example Output:
?- belongs(5,[2,8,12,5,6,9,7]).
true ;
false.
?- belongs(5,[2,8,13,5,6,9,5]).
true ;
true ;
false.
?- belongs(9,[2,8,13,5,6,9,5]).
true ;
false.
?- belongs(21,[21]).
true ;
false.
?- belongs(21,[]).
false.
?- belongs(5,[2,8,13,5,6,9,3]).
true ;
false.

Note: Before testing your .pl file you have to go to File>Save buffer, and then, Compile>Compile buffer

Wumpus [60 minutes – Exercises]

Task 3:

In this environment, an agent is given the task of navigating a 4 × 4 grid of squares. Some of these squares contain danger in the form of pits and a creature we shall refer to as the wumpus (which will gladly terminate the agent on sight). Not only is the agent unaware of which squares contain danger and which do not, but it is also unaware of how many pits can be expected to be found on the board (this number varies per game). The only information awarded to the agent are the contents of the square it currently resides in and sensory information that provides hints as to what dangers the agent can expect to find in the squares adjacent to it; if a pit or wumpus is in an adjacent square, the agent will be provided with a "wind" or "stench" sense, respectively. The most common methods of programmatically relaying the knowledge required to successfully navigate such an environment generally fall into one of two categories (though not necessarily exclusively): logical or probabilistic. The first of which being: How do we model our understanding of this environment in such a way that we can easily update the model given new information? And the second of which being: How can our agent best utilize this model to make decisions in the face of uncertainty? We shall address both of these challenges with the help of probability theory, a modest amount of graph theory, and the design of a utility function that makes sense of the final model and decides on an appropriate action

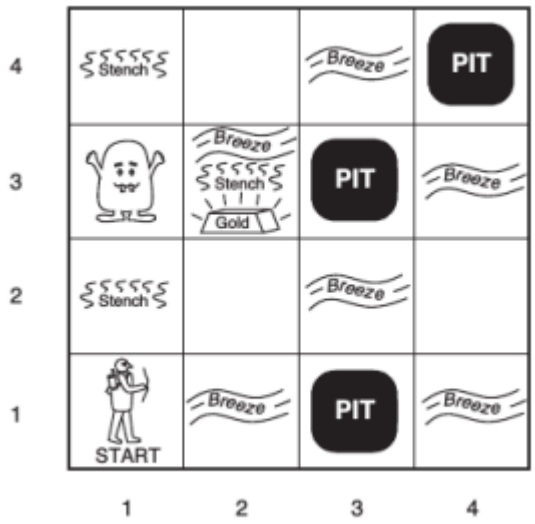


Figure 1: Wumpus world environment

The wumpus world is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the terrible wumpus, a beast that eats anyone who enters its room. Some rooms contain bottomless pits that will trap anyone who wanders into these rooms. The only mitigating feature of this bleak environment is the possibility of finding a heap of gold. Environment The environment is a 4 × 4 grid of rooms. The agent always starts the game in a corner square. The locations of the gold and the wumpus are chosen randomly, with uniform distribution, from the remaining squares. In addition, each remaining square has a 0.2 probability of being a pit. Figure 1 illustrates the typical features of a wumpus world environment, as we have just described it. Sensors If a square is adjacent (not diagonal) to a pit, the agent will sense a "Breeze" when at this square. If the square is adjacent to a wumpus, the agent will sense a "Stench." It is important to emphasize that the agent is only aware of the contents of the square that it currently resides in.